An Investigation of a Game Generator Tool to Teach Recursion

by

Jecton Tocho Anyango

Thesis Presented for the Degree of DOCTOR OF PHILOSOPHY in the Department of Computer Science, Faculty of Science, UNIVERSITY OF CAPE TOWN, South Africa

February 2022

Supervised by

Hussein Suleman

DECLARATION

I, hereby certify that this thesis has been written by me. I understand the meaning of plagiarism and declare that all the work in the thesis, other than which is properly acknowledged, is my own.

I further declare that the work reported in this thesis is the record of work carried out by me and that this thesis has not been submitted in any previous application for examination or to any other university or institution for any other degree or examination.

Date: 12/02/2022 **Signature of candidate**. Signed by candidate

(Jecton Tocho Anyango)

DEDICATION

*********To my immediate family********

To my beloved wife and companion Joyce Lirunjiro Luvuga and my three children Wesley Anyango Tocho, Rosley Iminza Tocho and Martin Luvuga Tocho for their constant assurance, love and support.

**********To the rest of the family*******

To my father Meshack Anyango Omoro from whom I drew the inspiration to pursue my PhD. studies.

To my mother Prescila Lwegado for her motherly love and care.

To my brothers and sisters for their unwavering support.

ABSTRACT

For many beginners, learning to program has proved to be difficult, in particular, learning the topic of recursion. Serious games have shown much promise in education, including in the teaching of programming. However, the adoption rate in mainstream teaching still remains low. One reason given for this is the lack of game authoring tools to support educators with little or no game programming skills. Moreover, developing educational games is difficult, time consuming and expensive. One way to address this problem is to use domain-specific game generators to create customised games as needed. The proposal in this thesis is that supporting higher education teachers with a game generator tool could potentially aid them to effectively create serious games. The aim was to investigate the use of a game generator tool to aid CS instructors to easily generate instances of educational games to teach recursion.

A User Centred Design (UCD) methodology was adopted to develop a prototype game generator tool called the Recursive Game Generator (RGG). Preliminary needs assessment and requirements analysis studies were conducted and conceptual design principles investigated. Four empirical studies were conducted to evaluate the prototype. Three were designed to evaluate the effectiveness of the prototype in supporting CS teachers to create games to teach novices the recursion topic while one evaluated the educational potential of the generated games from the lens of students.

Eight conceptual design principles were proposed that could guide the development of serious games that cater to diversity. It was also found that CS educators and trainees found the prototype useful, easy to use and learn; and were satisfied with the tool's effectiveness and efficiency, recommending its adoption. Students'feedback showed that the generated games had educational potential for learning programming.

The proposed conceptual design principles are insightful as they add new knowledge in the field of serious games design. Moreover, the positive empirical findings and user experiences suggest that such a higher-order tool has the potential to increase the adoption of serious games in programming education, and broadly meet the needs of a diverse audience of instructors and students.

ACKNOWLEDGEMENT

This thesis would have not been accomplished without the valuable financial support from the Hasso Plattner Institute for Digital Engineering (HPI), the National Research Foundation of South Africa and the University of Cape Town. I will forever remain grateful.

Special gratitude goes to my supervisor Hussein Suleman for his professional guidance through out my doctorate research work. One immediate thing I quickly learnt from him was academic writing - the need to present your arguments in a short and clear manner. His encouragement and patience when I faced challenges, especially during the Covid-19 pandemic, was enormous. Above all, Hussein has carefully nurtured me to become an independent scholar - something I will forever cherish.

I express my sincere thanks to all Computer Science teachers from higher learning institutions from Kenya, South Africa and other countries who took part in my evaluation studies. I want to recognise the trainee teachers from Kenyatta University and programming students from the University of Cape Town (UCT) who also took part in the evaluations. Appreciation also goes to Kandiri of Kenyatta university for helping in organising lab experiments with trainee teachers. To the entire Kenya Methodist University I am grateful. In particular, I extend my thanks to Chao Mbogho - my academic mentor.

I extend my appreciation to the team behind the conceptualisation, design, development and deployment of the prototype evaluated in this thesis. I want to acknowledge Enock Mbewe for the support during hosting with Amazon Web Services (AWS).

A special thank you to Gary Stewart and Aslam Safla (programming instructors from the department of computer science at UCT) for their insightful feedback during the iterative development.

Last but not the least, I would like to thank the entire membership of the ICT4D Laboratory and the Digital Libraries (DL) research group at UCT. The discussions we had and your critique shaped my thesis in one way or another.

TABLE OF CONTENTS

ABSTRACT i
ACKNOWLEDGEMENT ii
LIST OF FIGURES x
LIST OF TABLES xii
LIST OF TERMS AND ABBREVIATIONS xiv
LIST OF PUBLICATIONS xvii

1 Introduction **1**
1.1 Background 1
1.2 Problem statement 3
1.3 Motivation 3
1.4 Research Questions 5
1.5 Scope 6
1.6 Research Design 7
1.7 Main contributions 8
1.8 Organisation of Thesis 8

2 Related Work **10**
2.1 Overview 10
2.2 Programming Difficulty 10
2.2.1 Difficulty by programming languages and concepts 10
2.2.2 Difficulty by programming issues / problems 11
2.3 Recursion Difficulty 13
2.4 Game Based Learning Efforts and Conceptual Game Design 14
2.4.1 Games - based learning (GBL) efforts 15
2.4.2 Conceptual game design 18
2.5 Barriers to GBL Adoption 24

2.6 Game Generators . . . 26
2.6.1 Procedural content generation . . . 26
2.6.2 Commercial game generators . . . 26
2.6.3 Serious game generators . . . 27
2.6.4 Empirical evaluation . . . 29
2.6.5 Analysis of the evaluation studies . . . 32
2.7 Summary . . . 33

3 Understanding User Needs and Requirements 38
3.1 Introduction . . . 38
3.2 Study 1: Preliminary needs assessment . . . 38
3.2.1 Methods . . . 39
3.2.2 Result 1: Demographic Information . . . 41
3.2.3 Result 2: Programming issues . . . 41
3.2.4 Result 3: Difficult Introductory Programming (CS1) Topics . . . 42
3.2.5 Discussion . . . 43
3.3 Study 2: Requirements Analysis . . . 43
3.3.1 Methods . . . 45
3.3.2 Finding 1: Game Adoption Barriers . . . 46
3.3.3 Finding 2: Teaching Practices . . . 47
3.3.4 Finding 3: Teaching Analogies and Design Ideas . . . 49
3.3.5 Finding 4: Design for diversity . . . 50
3.3.6 Emerging conceptual design principles . . . 53
3.4 Study 3: Three visual ways to teach recursion . . . 56
3.4.1 Methods . . . 56
3.4.2 Student Interviews . . . 56
3.4.3 Visual Platforms Design . . . 56
3.4.4 Evaluation . . . 60
3.5 Summary . . . 61

4 Prototype Design and Development 63
4.1 Design Goal . . . 63
4.2 Design theories . . . 63

4.2.1 Complexity theory . . . 64
4.2.2 Differentiating interfaces theory . . . 65
4.2.3 Design theories choice . . . 65
4.3 Design Methods . . . 65
4.3.1 User participation . . . 67
4.3.2 Contextual inquiry . . . 68
4.3.3 Iterative design . . . 68
4.4 Prototype Development . . . 70
4.4.1 Review Meeting . . . 70
4.4.2 Technologies and the Architecture . . . 70
4.4.3 The Prototype Game Generator . . . 71
4.5 Game generation . . . 74
4.6 Generated Games and Features Designed . . . 77
4.6.1 First Iteration games . . . 77
4.6.2 Second and Third Iteration Games . . . 78
4.6.3 Features of the Generated Games . . . 79
4.7 Summary . . . 85

5 Experimental Design 86

5.1 Experiment One . . . 87
5.1.1 Purpose . . . 87
5.1.2 Participants . . . 87
5.1.3 Tasks . . . 87
5.1.4 Procedure . . . 88
5.1.5 Data collection Methods . . . 88
5.1.6 Criteria Used to Answer Research Question 2 . . . 89
5.1.7 Task difficulty . . . 89
5.1.8 Analysis and Presentation . . . 90
5.2 Experiment Two . . . 91
5.2.1 Purpose . . . 91
5.2.2 Participants . . . 91
5.2.3 Procedure . . . 92
5.2.4 Criteria Used to Answer Research Question 2 . . . 92

5.2.5 Data collection and Analysis . . . 93
5.3 Experiment Three . . . 93
5.3.1 Purpose . . . 93
5.3.2 Participants . . . 93
5.3.3 Materials . . . 94
5.3.4 Tasks . . . 94
5.3.5 Criteria Used to Respond to Research Question 2 . . . 95
5.3.6 Procedure . . . 95
5.3.7 Data collection and analysis . . . 98
5.4 Experiment Four . . . 99
5.4.1 Purpose . . . 99
5.4.2 Participants . . . 99
5.4.3 Materials . . . 99
5.4.4 Tasks . . . 100
5.4.5 Procedure . . . 100
5.4.6 Criteria Used to Respond to the Third Research Question . . . 101
5.4.7 Data collection and analysis . . . 102
5.5 Summary . . . 103

6 Results and Discussion 104

6.1 Introduction . . . 104
6.2 Experiment One . . . 104
6.2.1 Demographics . . . 104
6.2.2 Task difficulty . . . 105
6.2.3 Task success . . . 105
6.2.4 Estimated time on tasks . . . 106
6.2.5 Usefulness . . . 107
6.2.6 Ease of Use . . . 108
6.2.7 Ease of Learning . . . 108
6.2.8 User Satisfaction . . . 109
6.2.9 Final user comments . . . 109
6.2.10 Discussion . . . 111
6.3 Experiment Two . . . 112

6.3.1 Demographics . . . 113
6.3.2 Usefulness and suitability of the prototype . . . 113
6.3.3 Prototype's most useful features . . . 114
6.3.4 Final comments . . . 115
6.3.5 Discussion . . . 116
6.4 Experiment Three . . . 117
6.4.1 Demographics . . . 117
6.4.2 Scale mean scores . . . 118
6.4.3 Pragmatic Quality . . . 119
6.4.4 Hedonic Quality Identity (HQ-I) . . . 119
6.4.5 Hedonic Quality Stimulation (HQ-S) . . . 120
6.4.6 Attractiveness . . . 120
6.4.7 Qualitative comments . . . 120
6.4.8 Discussion . . . 122
6.5 Experiment Four . . . 123
6.5.1 Demographics . . . 123
6.5.2 CS1 Students'Game Play Experiences . . . 124
6.5.3 Finding 3: CS1 Students'Perception of Played Games . . . 127
6.5.4 Finding 4: Rankings of most useful design features . . . 128
6.5.5 Open ended questions responses . . . 128
6.5.6 Summary . . . 134
6.6 Comparative analysis of participant cohorts results . . . 135

7 Conclusions 137

7.1 Summary of thesis . . . 137
7.2 Summary of the study findings against research questions. . . . 138
7.2.1 RQ 1: What conceptual attributes must a generator tool take into account in order to generate a game that can teach recursion? . . . 138
7.2.2 RQ 2: How effective is the use of a generator tool in creating customised game instances to teach recursion . . . 139
7.2.3 RQ 3: What is the suitability and educational potential of the generated games to enable students to learn the topic of recursion? . . . 141
7.3 Research Implications . . . 144

7.3.1 Theoretical Implication 144
7.3.2 Practical implications 144
7.4 Research Limitations 145
7.4.1 Scope 145
7.4.2 Sample size 146
7.4.3 Student learning gain 146
7.5 Future Research and Reflection 147
7.5.1 Future Research 147
7.5.2 Reflection 149
REFERENCES 149

Appendices

Appendix A Preliminary study **181**
A.1 Expert interview guide 181
A.2 Survey questionnaire 182

Appendix B User requirements study **185**
B.1 Interview guide for teachers 185
B.1.1 Teaching recursion experience 185
B.1.2 Recursion educational games and game generator tool design requirements 185

Appendix C Experiments 1, 2 , 3, and 4 Questionnaires and Ethics Clearance) **187**
C.1 Experiment 1 (Pre-test questionnaire) 187
C.2 Experiment 1 (Post-test questionnaire) 188
C.3 Experiment3 Questionnaire - Part A: Demographics 190
C.4 Experiment3 Questionnaire - Part B: User experience when creating a game with a static game example (Prototype with a static game example) . 191
C.5 Experiment3 Questionnaire - Part C: User experience when creating a game with a dynamic game example (Prototype with a dynamic game example 192
C.6 Questionnaire for Experiment 4 with CS1 Students 193

C.7 Experiment 4 Game Experience Questionnaire 194
C.8 Experiment4 Questionnaire - Parts C , D and E 195
C.9 Experiment 1 Positive Comments Open Ended Responses Raw Data . . . 196
C.10 Experiment 1 Negative Comments Open Ended Responses Raw Data . . . 197
C.11 Experiment 1 Final Comments Raw Data 198
C.12 Experiment 2 Open Ended Responses Raw Data 199
C.13 Experiment 3 Open Ended Responses Raw Data 200
C.14 Experiment 4 Open Ended Responses Raw Data 201
C.15 Ethics Approval for Preliminary Needs Assessment Study 202
C.16 University of Cape Town Ethical Clearance 203
C.17 The National Commission for Science, Technology and Innovation (Nacosti) Research Permit 204
C.18 Kenyatta University Permission to Conduct Research 205
C.19 Ethics Approval to Conduct Research with UCT CS1 Students 206
C.20 Ethics Approval to Access UCT Students 207
C.21 Game generation step 1 208
C.22 Game generation step 2 209
C.23 Game generation step 3 210
C.24 Game generation step 4 211
C.25 Game generation step 5 212
C.26 Game generation step 6 213
C.27 Game generation step 8 214
C.28 Game generation step 9 215
C.29 Trophies and health points as reward 216

LIST OF FIGURES

2.1 The Elemental Recurrence Game - Level 2 [48] 16
2.2 Cargo-Bot Game Play Scenario [255] 17
2.3 LightBot Game Play Scenario [2] 18
2.4 Conceptual Framework for Educational Games Design [70] 21
3.1 Summary of conceptual design principles 55
3.2 Visual coding User Interface 58
3.3 Dropping commands in the drop zones 59
3.4 Main stack simulation User Interface 59
3.5 Full built stack for recursive power 60
3.6 Power of a number use case 60
3.7 Factorial of a number use case 60
4.1 Paper prototype 1 67
4.2 Paper prototype 2 67
4.3 The Recursive Game Generator (RGG) Architecture 71
4.4 The Recursive Game Generator - RGG 72
4.5 Registration interface 72
4.6 Help Feature 73
4.7 Generated game download 73
4.8 Level concept/ description/ question configuration 76
4.9 Downloaded Items 76
4.10 First Iteration Game 77
4.11 DnD game instance 79
4.12 Mushroom Picker game instance 82
4.13 Mushroom picker game Level 3 play interface 83
4.14 Visualisation, IDE and Pedagogy features 84
5.1 Group 1 Participants 97

5.2 Group 2 participants 97
6.1 Task Difficulty Rating 106
6.2 Estimated Time on Task Rating 107
6.3 Perceived Tool Usefulness 108
6.4 Perceived Ease of Use 108
6.5 Perceived Tool Ease of Learning 109
6.6 Perceived User Satisfaction 110
6.7 Responses to Additional Usefulness items 114
6.8 Attrakdiff dimensions average values 118
6.9 Pragmatic and Hedonic Quality Identity Word Pairs Mean Values 120
6.10 Hedonic Quality Stimulation and Attractiveness Word Pairs Mean Values 121
6.11 Perceived Competence 126
6.12 Perceived tension 127
6.13 Perceived Positive Effect 128
6.14 Perceived Negative Effect 129
6.15 Students'perceptions of played games on additional items 129

LIST OF TABLES

2.1 Analysis of Difficult Programming Topics According to Instructors and Students . . . 34
2.2 Educational Games Designed to Teach Recursion (Elem= Elemental, CB= CargoBot, LB = LightBot, PYR = Program your Robot, SC = Saving Cera) . . . 35
2.3 Constructivism and behaviourism pedagogical theories and conceptual Serious Games (SGs) design (Adopted from [99, 158, 277]) . . . 36
2.4 Summary of Pedagogical Conceptual Design Considerations . . . 36
2.5 Summary of Game Play Conceptual Design Considerations . . . 37
2.6 Summary of Empirical Evaluation Studies . . . 37
3.1 Demographic information . . . 42
3.2 Issues making learning programming difficult for novices . . . 42
3.3 A summary of results on difficult CS1 topics . . . 43
3.4 Customisable Game Design Attributes . . . 52
3.5 Analysis of student opinions about their challenges when learning the recursion topic . . . 57
3.6 A summary of some student design suggestions . . . 58
3.7 A summary of system requirements . . . 62
4.1 Summary of four philosophy driven instructional design paradigms (adapted from [269] in [21]) . . . 68
4.2 Demographic information . . . 69
4.3 A mapping of conceptual design principles, design theories and the designed prototype features . . . 74
4.4 Alignment of the Mushroom picker game levels to pedagogy in CS1 . . . 80
4.5 Alignment of the DnD game levels to pedagogy in CS1 . . . 81
6.1 Demographic information of Experiment one . . . 105

6.2 Number of participants successful on the tasks under various task success levels . . . 106
6.3 Demographic information of Experiment two . . . 113
6.4 Ranking of the prototype's perceived useful features . . . 115
6.5 Top 8 themes from final general comments . . . 116
6.6 Demographic information of Experiment three . . . 117
6.7 Mean values . . . 118
6.8 GEQ statistics for the Mushroom Picker and DnD Games, N = 20, GEQ Scale Ranges from 0 to 4 (adapted from [183] . . . 125
6.9 Reliability statistics for the Mushroom Picker and DnD Games, N=20 . 130
6.10 Ranking of most useful design features of played games . . . 130
6.11 Detailed thematic analysis of RGG's effectiveness and potential . . . 131
6.12 Comparative Analysis of Participant Cohorts Results . . . 136
7.1 Summary of of Empirical Evaluation studies . . . 143

LIST OF TERMS AND ABBREVIATIONS

ACM Association of Computing Machinery

ADRI Approach Deployment Result Improvement

ARSGs Augmented Reality Serious Games

ATMSG Activity Theory Model for Serious Games

ATT Attractiveness

AWS Amazon Web Services

CFW Conceptual Framework for Educational Games Design

CLT Cognitive Load Theory

CS Computer Science

CS1 Introductory Programming

CSE Computer Science Education

CSOs Campus Security Officers

CSS Cascading Style Sheets

DFS Depth First Search

DL Digital Libraries

EGG Extensible Graphical Game Generator

GBL Games - based learning

GCD Greatest Common Divisor

GEQ Game Experience Questionnaire

HCI Human Computer Interactions

HQ-I Hedonic Quality Identity

HQ-S Hedonic Quality Stimulation

I/O Input Output

ICT Information and Communication Technology

ICT4D Information and Communications Technology for Development

IDE Integrated Development Environment

IEEE Institute of Electrical and Electronics Engineers

IRPG Instruction Right Place Game Generator

IS Information Systems

ITiCSE Innovative Technologies in Computer Science Education

JS Java Script

KU Kenyatta University

LG-GM Learning Mechanics - Game Mechanics

LPG Learning version of Pacmac Game Generator

MLGs Mobile Learning Games

NACOSTI National Commission for Science, Technology and Innovation

PRA Pragmatic quality

QoLE Quality of Learning Experience

RETAIN Relevance Embedding Transfer Adaptation Immersive Naturalisation

RGG Recursive Game Generator

SCLOs Storytelling Complex Learning Objects

SGs Serious Games

SPSS Statistical Package for Social Sciences

STEM Science, Technology, Engineering and Mathematics

SUS System Usability Scale

TAM Technology Acceptance Model

TAs Teaching Assistants

TCA Thematic Content Analysis

TEI Technology Enabled Interventions

Trainees Trainee Teachers

UCD User Centered Design

UCT University of Cape Town

UD Universal Design

UI User Interface

USE Usefulness Ease of Use

UX User Experience

WEEV Writing Environment for Educational Video Games

LIST OF PUBLICATIONS

Some of the work presented in this thesis has appeared in the following publications:

1. Anyango, J. and Suleman, H. Supporting cs trainee teachers with game authoring tools. 2021.
2. Anyango, J. T. and Suleman, H. Teaching programming in kenya and south africa: What is difficult and is it universal? In Proceedings of the 18th Koli Calling International Conference on Computing Education Research, pages 1–2, 2018.
3. Anyango, J. T. and Suleman, H. Designing Programming Games for Diversity in Teaching Introductory Programming. In: Wells G., Nxozi M., Tait B. (eds) ICT Education. SACLA 2020. Communications in Computer and Information Science, vol 1518. Springer, Cham. https://doi.org/10.1007/978-3-030-92858-22
4. Anyango, J. T. and Suleman, H. Supporting cs1 instructors: Design and evaluation of a game generator. In Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1, pages 115–121,2021.

Additionally, some aspects of the work presented in Chapter 3 are as a result of a Honours Project titled - "BBRV - Three Visual Approaches to Aid the Teaching of Recursion", supervised by the author. The final student reports on visual coding, visual simulation and visual simulation approaches can be found using the link: https://projects.cs.uct.ac.za/honsproj/2019/.

CHAPTER 1

Introduction

1.1 Background

Learning programming has proved to be difficult among most students in higher education [124, 137]. In particular, novices struggle with learning the recursion topic in Introductory Programming (CS1) courses [146, 233]. Other CS1 topics that learners equally find difficult include arrays, loops, pointers and Abstract Data Types (ADT). Owing to the programming difficulty, high failure and drop out rates have been reported in CS1 [124, 119, 32, 34, 199, 205, 272] world-wide. In their study on failure rates in CS1 in 2007, Bennedsen and Caspersen [34] reported an average failure rate of 33%. On this issue, their work and that by Watson and Li [272] are the most widely cited by CSE community papers with the impression that CS1 failure rates are high. However, recently there was an observation that some of the citing papers have misrepresented what was originally written by the authors [138].

Bennedsen and Caspersen replicated their study in 2017 and reported an average failure rate of 28% in CS1, noting an improvement. Comparing this rate to 42% - 50% failure rate in college algebra in the US, the authors concluded that 28% does not appear alarmingly high. Nonetheless, a working group on Innovative Technologies in Computer Science Education (ITiCSE) found that pass rates in introductory programming courses appear to average about 75% and that there is some evidence that they are at the low end of the range of pass rates in Science, Technology, Engineering and Mathematics (STEM) courses [239]. Meanwhile, there are still major concerns internationally among academicians that educators cannot teach an essential skill like programming more successfully and reliably to students after over forty years of teaching the course [162]. In fact, introducing students to computing still remains one of the grand challenges among educators in the community [35]. This is coupled with the challenge of designing more inclusive and effective learning environments and instructional methods for CS1 [35, 176]. This problem has prompted the Computer Science Education (CSE) community to call for improvement in CS1 pass rates and alternative approaches of improving student learning in CS1 courses [239].

In most universities, many students are annually registered in CS1 [119, 230]. The course enrols learners from diverse backgrounds academically, economically, culturally, as well as in terms of learning and motivational needs [124]. For example, the course attracts non Computer Science (CS) majors from Information Systems (IS), Actuarial Science, Mathematics, Statistics, Engineering, and Business disciplines, among others. While some have been exposed to computing technology before joining higher education, others have not. This diversity presents both teaching and learning challenges. In fact, within the CSE community, research has shown that programming novices and teachers continue to face serious difficulties and challenges [124, 146].

Some researchers have reported that teachers find it challenging to teach the CS1 course [164]. Three teacher related challenges have been identified as the: (i) methods and teaching tools; (ii) scale problems (large classes with diversity); and (iii) low student motivation and engagement [164]. Given the programming difficulty and the need for innovative methods of improving student learning and pass rates, especially in CS1, some researchers have proposed Games - based learning (GBL) as an alternative approach [2, 48, 113]. Prensky defines GBL as the design of game applications with clear learning outcomes [209]. A serious game is an educational game designed to teach people about a particular subject, develop concepts, or assist in learning a skill or changing player attitudes [62]. Other scholars have also defined serious games as games designed for use in contexts other than entertainment or to achieve goals other than entertainment, such as education, health, cultural heritage etc [120, 165]. They are games designed to educate, train, and inform [117]. An educational game is a game that is used as an educational tool that provides interactive and appealing activities that attract students'interest for learning [1]. These definitions are adopted in this thesis. Some examples of serious games that have been designed to teach programming in computing education include Light-Bot[1] [2], Elemental the Recurrence [48], Alice [58] and Cargo-Bot[2] [255].

For the purpose of the user studies conducted in this thesis, the adopted definition of usefulness is the one proposed by MacDonald and Atwood [141]. According to Macdonald and Atwood [141, p. 2], usefulness is defined '*as the extent to which a system's functions allow users to complete a set of tasks and fulfill specific goals in a particular context of use*'.

[1] https://lightbot.com/hour-of-code.html
[2] http://i4ds.github.io/CargoBot/?state=1

1.2 Problem statement

Although GBL has been proposed as one of the alternative approaches to deal with programming difficulty, the adoption rate still remains low in mainstream teaching [253, 85]. Some of the reported barriers to the adoption of GBL in mainstream teaching and learning include the (i) fact that current serious games designed to teach programming do not support diversity as well as enable educators to customise them to their students needs [148], (ii) critical limited number of serious games that are aligned with curriculum or those that can be easily adapted by teachers to fit their learning goals [228, 85], (iii) lack of game programming skills by teachers [87, 3], (iv) the complex nature of the process of developing serious games and high development costs involved [87, 261, 262, 3], (v) lack of instructor-oriented serious game authoring tools to support teachers with limited game programming skills [111, 253], (vi) difficulty of harmonising educational value and fun [261], (vii) ill prepared teachers [29], and (viii) limited time in the curriculum [87, 261, 3], among others. A detailed discussion of these barriers is presented in Section 2.5 of the literature review.

Although serious games have shown much promise in education, including in the teaching of programming, the problem is that instructors often do not have the specialised skills to create such games [253, 85, 87, 96, 261, 281, 111, 134]. This hinders the adoption of GBL in mainstream teaching. Moreover, there is lack of serious game authoring tools that could support programming teachers with little or no game programming skills [111, 253, 87, 261]. Meanwhile, developing serious games is time consuming, difficult and expensive [87, 261].

1.3 Motivation

Literature on programming difficulty can be divided into two categories: (i) learning programming and (ii) teaching programming. The first focuses on students - their learning, misconceptions and difficulties they face when learning to program. The second is on educators - their teaching strategies, instructional tools and the challenges they face when teaching. This thesis primarily targets the second ignored category. Serious games have shown much promise in education. However, the adoption rate in mainstream teaching is still low. One of the reasons given for the low adoption rate is that teachers lack specialised skills to develop games [253, 85, 87, 261, 262, 111, 134]. Meanwhile, there are limited game generator tools that can support teachers to easily create games. Educators often look to technology to support their teaching. To this end, some scholars have suggested Technology Enabled Interventions (TEI) such as game authoring platforms [150, 204]. However, current game authoring tools have some problems. One of them is that most tools and technology platforms such as the

Creator[3], Game Maker [4], and Mission Maker[5] are commercial hence target specialised developers or game programmers [263]. In addition, current tools are not designed for use by instructors as they lack pedagogy [261]. Moreover, they support few game play alternatives such as challenges and activities that can be inserted in educational games [282, 240, 261]. Marchiori et al. [150, p. 1] reported that usability tests of several current game authoring tools with educators suggest that they are complicated. The authors further alluded that the importance of *"the authoring workflow suitable for educators as target users has not been sufficiently addressed"* [150, p. 1]. In other words, the authoring workflow is complicated.

Another problem is that current authoring tools do not support game design processes from domain experts's (educators's) perspectives [240, 4, 263] that is there is limited design role of educators [240, p. 1]. Ahmad et al. note that generally current game authoring tools *"do not implicitly encapsulate any structured process of designing game play from knowledge content of a domain"* [4, p. 1]. Indeed, there are limited works on end user development for serious games [168]. Furthermore, little attention has been given to studies that demonstrate the effectiveness of current game authoring tools and of the authored games [168]. In particular, only few works have explored game authoring tools for programming games as demonstrated in the analysis in Section 2.6.3.

In view of these, some scholars have underscored the need to (i) create affordable serious games that are aligned to the curriculum, (ii) involve educators in the design process, (iii) design game authoring tools that embrace pedagogy and those that can be used by non programmers [261, 263], and (iv) evaluate the effectiveness of the authoring tools and the generated games with educators and students [195] to demonstrate their effectiveness, suitability and educational value [85, 195].

To address the problem of lack of specialised game development skills, teacher-oriented authoring tools that can allow educators with basic computer skills to easily create and customise their own educational games have been suggested [85, 134]. It is hoped that this will also address the problem of the critically low number of serious games aligned to the curriculum [85, 134]. To add to the current knowledge in this area, this thesis therefore investigated a game generator tool to support higher education instructors to easily create instances of serious games. A game generator is a software platform that allows non-experts to generate games according to their needs or specifications [72]. The central argument in this thesis is that supporting CS educators with a game generator tool could potentially aid them to easily and effectively create serious games to teach programming. The thesis statement is therefore: *The effective use of a*

[3]https://www.thegamecreators.com
[4]https://gamemaker.nl
[5]https://immersiveeducation.com/missionmaker

game generator tool by CS educators can aid them to create custom serious games to teach programming.

The author is a practicing educator in one of the universities in Kenya. As in other higher learning institutions, many diverse students are registered in the CS1 course in the author's university but a large proportion fail or drop out. Given this experience, the author developed an interest to investigate other fun ways to teach abstract concepts in the course.

To make a contribution towards addressing the problem of lack of authoring tools to support domain experts with little or no game programming skills, this thesis proposed a serious game authoring tool. This was intended to make it easy for teachers with basic computer skills to easily create serious games [134, 72, 263]. It was hoped that findings of this study would demonstrate the: (i) effectiveness of using such a higher order toolkit by programming instructors, and (ii) educational potential of the generated games for learning. Effective use of teaching and learning materials or tools has been cited as an important factor in tackling the difficulty in teaching and learning computer programming [50].

1.4 Research Questions

To address the research problem, the broad research question is: *"To what degree can the use of a game generator tool create customised games that teach recursion"*? In particular, this thesis answers three specific research questions:

1. *What conceptual attributes must the generator tool take into account in order to generate games that can teach recursion*?
   The rationale of this question was to guide the methodical process of identifying key conceptual attributes (learning and game attributes) necessary to stimulate the design and development of a prototype game generator tool that could create playable game instances to teach the recursion topic. To address this question, a requirements study was conducted. CS1 teachers were interviewed to give design insights for a prototype game generator tool capable of generating games that could teach the recursion topic to diverse CS1 students. This culminated in eight conceptual design principles as discussed in Chapter 3.

2. *How effective is the use of a generator tool in creating customised game instances to teach recursion*?
   Through empirical evidence, this question was meant to help elicit feedback from the users to provide evidence on the effectiveness of using the prototype game generator tool by CS teachers and Trainee Teachers (Trainees) in creating games to teach the recursion topic. Effectiveness was seen in terms of perceived toolkit

usability, User Experience (UX), additional benefits, and the educational potential of the generated games for learning programming. CS1 teachers evaluated the prototype in usability studies. Collected data was analysed to measure the prototype's: usefulness, ease of use, and learnability. Users'satisfaction with the prototype was also measured. Meanwhile, data collected from CS trainee teachers'appraisal of the prototype was analysed to measure: the Pragmatic quality (PRA), Hedonic Quality Identity (HQ-I), Hedonic Quality Stimulation (HQ-S) and Attractiveness (ATT) dimensions to give an indication of the prototype's effectiveness using the Attrakdiff UX metrics [94].

3. *What is the suitability and educational potential of the generated games to enable students to learn the topic of recursion*?
   It is important to note that the work in this thesis is about a game generator tool and not a game. However, the rationale of this question was to further elicit feedback from students to provide evidence that the generated games could help students to learn the topic of recursion. The question is significant as it elicits feedback that demonstrates that the games are valid from the lens of learners. Three metrics were used to measure the suitability and the educational potential of the games, namely the: (i) user experience of CS1 students after playing the games created from the prototype game generator tool; (ii) game design features CS1 students found most useful; and (iii) CS1 students'overall opinions about the educational potential of the generated games to enable them to learn the difficult topic of recursion.

1.5 Scope

With regard to CS1 topics, this study was limited to the topic of recursion as a case study. Although a number of problematic CS1 topics were highlighted in Section 1.1, the choice was motivated by the fact that the recursion concept is powerful and core in CS [243, 161] as emphasised in the introductory programming courses of every CS curriculum [227, 236] yet students struggle to understand it [276]. Additionally, a recent review of gamification for learning programming fundamentals found out that most serious games focus on conditionals, iterations, variables, and primitive data types, with least focus on concepts such as recursion, linked lists, referencing, and abstract data types [236]. The review considered games designed between 2015 and 2020. Concerning evaluation, it was not within the scope of this thesis to measure student learning gains or outcomes. Instead, the evaluation focused on two aspects. The first looked at the effectiveness of using a prototype game generator tool called the Recursive Game Generator (Recursive Game Generator (RGG)) by CS educators and Trainees in creating customised game instances to teach the topic of recursion. The second evaluated the

generated games with students to demonstrate the suitability and educational potential of the generated games from the lens of students.

1.6 Research Design

The research design adopted for this thesis was the mixed methods approach. This was because it puts together the strengths of qualitative and quantitative methods [80]. Both qualitative and quantitative data were collected from CS1 teachers, Trainees and students and analysed. The study was conducted in four phases. Phase one involved a preliminary needs assessment study conducted with CS1 teachers to identify the problem and user needs. This was achieved through an online survey. CS educators identified CS1 topics deemed difficult for novices. Findings of this study informed the choice of the recursion topic as a case study to test the prototype game generator tool developed in this thesis. Recursion was chosen because it was considered one of the most difficult CS1 topics among others like Abstract Data Types, Pointers and Arrays.

To address the first research question, online interviews were conducted with CS1 educators in the second phase. This provided in-depth insights about the conceptual design principles during the ideation stage. To gain further understanding of the conceptual design attributes, educator responses to open ended questions were analysed thematically. In addition, the author supervised three honours students from the department of Computer Science at the University of Cape Town (UCT). Three visual aspects of teaching recursion were investigated. Findings from these studies culminated in eight conceptual design principles and a user requirement document as summarised in Chapter 3. These outputs guided the third phase (design of the experimental prototype). This phase adopted the User Centered Design (UCD) methodology. UCD is a design *"approach based on the active involvement of users to improve the understanding of user and task requirements, and the iteration of design and evaluation"* [149, p. 1]. A prototype game generator toolkit - Recursive Game Generator (RGG) - was developed through three iterations with the help of two CS1 teachers from UCT as informants.

To answer the second and third research questions on the effectiveness of the proposed game generator tool and the educational potential of the generated games respectively, a combination of evaluation methods was used in the fourth phase. For the second research question, the methods used included usability surveys using standard user experience measurement tools such as the Usefulness Ease of Use (USE) questionnaire [135] and the Attrakdiff questionnaire [94]. In addition, educators provided final opinions about the prototype game generator tool. The USE questionnaire helped to measure the prototype's ease of use, ease of learning, and user satisfaction. The Attrakdiff questionnaire sought to provide additional insights on the summative evaluation of the user experiences regarding the usability, the HQ-I, HQ-S and ATT. For the third

research question, the Game Experience Questionnaire (GEQ) [103] was used to assess student experience with the played generated games. In additional, final comments gave further insights about students'sense of the educational value or potential of the played games.

1.7 Main contributions

This thesis contributes to the body of knowledge in GBL in CSE by investigating and evaluating a prototype game generator as a potential support tool for non-technical CS teachers. Non-technical in this thesis refers to those teachers who lack game programming skill. To practicing educators within the CSE community, the thesis offers a novel game authoring toolkit prototype. To the game development industry, the study proposes design principles that could guide development of future game generators and serious games. In a nutshell, this thesis contributes to the design, development and evaluation of an innovative game generator toolkit to support programming instructors in higher education. Specifically, the following five contributions are envisaged:

1. A proof of concept prototype game generator tool to support non-technical CS1 educators to create programming games to teach the topic of recursion.

2. Conceptual design principles to guide the design and development of a game authoring tool to teach the recursion topic to diverse novice students.

3. Empirical evidence of the usability of a prototype game generator tool by CS1 educators.

4. Empirical evidence of the potential and suitability of a prototype game generator tool to support CS trainee teachers during their teaching practice and immediately when they embark on teaching.

5. Empirical evidence of the effectiveness of the prototype by demonstrating the educational potential and suitability of the generated games to enable CS1 students to learn the recursion topic.

1.8 Organisation of Thesis

Chapter 2 discusses previous work in the research topic. First, a broad survey of programming learning difficulty among students is covered. This is followed by a review of studies specifically about recursion learning difficulty. Afterwards, previous works on GBL efforts, GBL adoption barriers and conceptual serious games design and development are synthesised. This is followed by an analysis of commercial and serious

game generators with emphasis on the evaluation studies. Finally, research gaps are analysed and a summary of the Chapter presented.

Chapter 3 attempts to understand the user needs and requirements. It summarises three studies done to achieve this aim. It starts by discussing a preliminary needs assessment study done to understand the problem and user needs. Next, it presents a user requirements study conducted to gain insights about the functional and non-functional system specifications as well as serious games conceptual design principles. Finally, it discusses an honours project (BBVR - Three visual ways to teach recursion) supervised by the author as a way of further understanding user requirements.

Chapter 4 presents the design and development of the prototype game generator tool called the RGG. First, it presents the design goal then analyses the theories that guided prototype design and development. In addition, it provides justification for the choice of the hidden complexities and differentiating interfaces theories. The Chapter further discusses the adopted UCD methodology, the prototype development, game generation, and the generated games and features designed.

Chapter 5 is on the experimental design adopted to evaluate the prototype game generator tool designed in this thesis and the generated games. It describes the purpose of each empirical study, participants, tasks performed, procedures followed, and data collection and analysis methods. For each experiment, it also discusses the criteria used to answer the second and third research questions.

Chapter 6 presents the results from the four empirical evaluations conducted in this thesis. Three with educators while using the prototype game generator tool and one with students while playing the generated games. First, findings of a usability study conducted with CS educators from Kenya and South Africa are presented and discussed. Second, results from a user experience study with CS educators from the wider CSE community are presented and analysed. Third, findings from a user experience evaluation with Trainees from Kenya are presented and discussed. Lastly, the Chapter summarises and discusses results from a game experience experiment with CS1 students from UCT - South Africa.

Chapter 7 is the last Chapter in this thesis. It concludes the research undertaken by first presenting a summary of the findings against the research questions. In doing this, it revisits the thesis aim and the research questions. Next, it synthesises the results and explains how the research questions were answered. It summarises the contributions and implications of the research findings. It also covers limitations and future development. Finally, it highlights suggested future research directions and the author's final reflection.

CHAPTER 2

Related Work

2.1 Overview

This Chapter presents the related work. The review broadly presents a survey of the body of work on CS1 learning difficulties, the attempted GBL efforts and game generators. Section 2.2 presents a broad overview of programming difficulty among students in higher education. This is followed by a discussion of the recursion topic difficulty in Section 2.3. Some GBL efforts and conceptual game design are discussed in Section 2.4. Section 2.5 is about barriers to games adoption in mainstream teaching. Existing game generators, their features and limitations are discussed in Section 2.6. Finally, Section 2.7 concludes the Chapter by presenting a case for a programming game generator tool to create games to teach introductory programming.

2.2 Programming Difficulty

Many studies have reported that learning programming is difficult for most students in higher education [107, 124, 159, 172, 202, 220, 221]. Some have pointed out how difficult it is to teach the "Nintendo" generation how to program given their low motivation levels [89]. Consequently, high failure and drop out rates have been reported in CS1 courses [34, 272, 119]. At the same time, the representation and participation of women in CS have remained low [78]. This situation has persisted notwithstanding the steady rise in job opportunities and demand for CS graduates by the industry [268]. In this thesis, the literature on studies that have investigated programming difficulty is considered under those that considered: (1) difficulty by specific programming languages and the abstract concepts; and (2) difficulty due to other issues/ problems.

2.2.1 Difficulty by programming languages and concepts

Studies under this category have explored difficult concepts/ topics according to students and instructors. Difficulty by programming language has also been considered by some works in this category. Falling in the first group is a recent study that inves-

tigated Chinese middle students' difficulties in learning to program in Python [212]. Results showed that students struggled with fundamental Python syntax and programming rules. Similarly, another research project in this category sought the views of students and tutors when using the C++ language on difficult topics [173]. The authors singled out pointers, virtual functions and dynamic allocation of memory as the difficult topics and proposed a visualisation tool. The study also ranked recursion among the top most difficult topics.

Lahtinen et al. surveyed over 500 students and teachers to find out difficult CS1 concepts with Java and C++ programming languages [124]. Results suggested that the most difficult programming concepts were recursion, pointers and references, abstract data types, error handling and using language libraries. In this study, findings on the difficult concepts was similar for both teachers and students. Regarding programming languages, the study found out that C++ was more difficult compared to Java.

Schulte and Bennedsen asked university, college, and high school instructors to rate the difficulty among 28 CS1 topics [234]. In line with the works by Milne et al.[173] and Lahtinen et al. [124], they also found that recursion, pointers and references are hard topics. Recursion was rated the most difficult topic in both universities and colleges. In addition, polymorphism was also considered difficult. According to this study, C++, Java and Pascal were the programming languages used by most teachers. However, no comparison was done to determine which of the three was the most difficult.

Bosse et al. carried out a study involving 16 semi-structured interviews with instructors on Python and C languages and diaries from 110 students on their CS1 learning difficulties [39]. Results indicated that the difficult topics were: variables; expressions; functions; Input/ Output functions; selection and repetition structures; uni and multi-dimensional arrays; and pointers and strings. Meanwhile, Dale surveyed 350 educators on the most difficult CS1 topics for beginners and reported control structures, Input Output (I/O), recursion and object orientation topics like inheritance, polymorphism, methods, classes, instance and static variables [57].

2.2.2 Difficulty by programming issues / problems

The second category of studies have not only looked at programming difficulty from the perspective of specific programming languages and (or) the abstract concepts/ topics but have also investigated issues (problems) that make learning programming hard. One such study showed that the course is difficult because of issues like: (i) lack of understanding how to design a program to solve a certain task; (ii) challenges in dividing functionality into procedures; (iii) challenges with finding bugs in programs; (iv) students overestimating their understanding; (v) lack of practice and application of learned concepts; and (vi) not understanding programming structure [124].

Another comprehensive review of literature on programming difficulty found out that students' misconceptions and other difficulties in CS1 are related to syntax, conceptual and strategic knowledge [211]. This was mainly attributed to syntax issues, poor mathematics knowledge, wrong mental models, students' prior knowledge and programming environments. Another recent study with 771 novice students found that some of the common challenges students encounter when transiting from one programming language to another include syntax, error messages, and the process of compilation [63].

Consistent with findings by Lahtinen et al. [124] and Qian et al. [211], Laporte and Laman found 10 base programming problems [128]. They grouped them into three components - namely: (i) process components - problems relating to basic steps in programming such as design, working constructs, and evaluation; (ii) cognitive components such as writing syntax, constructing mental models, and problem solving; and (iii) affective components - attitude and motivation aspects. Additionally, similar to other works [128, 44, 124], the study by Dale found two novice programmer issues: (i) problems involving higher order thinking like problem solving and design; and (ii) student lack of maturity and study skills [57]. The finding on problem solving difficulty was recently supported by another research with 113 students [129]. Results indicated *"that students do face a significant level of difficulty when applying problem solving skills in learning programming languages "* [129, p. 1]. The highest perceived difficulty was understanding a program without first coming up with it's algorithm [129].

Meanwhile, in an investigation of the reasons for high attrition rates for CS students, Beaubouef and Mason [32] reported: (i) poor maths and problem solving abilities; (ii) poorly designed CS1 courses; (iii) poor advising during and before college; (iv) lack of practice and feedback; and (v) untrained student graduate teachers, among others. In another study, when views of five experienced CS1 lecturers were sought on the issues that make programming difficult to students, respondents suggested lack of skills such as: (i) critical thinking; (ii) problem solving; (iii) design of algorithms; and (iv) debugging [44]. Further, lack of self confidence and misconceptions about computing were mentioned. Recently, another study was conducted with in-service teachers in attempt to find out what teachers perceive to be difficult in introductory programming [225]. Out of the issues that make programming challenging for novices, participants expressed that writing code and pseudo-code to solve a specific problem are the most difficult [225].

A survey of 158 students who had attended a first year course in programming at Reykjavik University (RU) in Iceland indicated that novice students found the task of dividing an activity into functions and classes and finding errors in a program as the most difficult [156]. Again more recently, a systematic review on teaching and learning introductory programming in higher education categorised novice programming students'difficulties into four [164]. They include: (i) problem formulation - problem

solving, abstract nature of programming, and algorithmic and logical reasoning; (ii) solution expression - programming language syntax, control structures, data structures, structure of code and others; (iii) solution execution and evaluation - debugging and tracing the execution; and (iv) student behaviour - time management, motivation and engagement, study skills, confidence and perceiving programming as difficult [164].

Table 2.1 presents a comprehensive analysis of CS1 difficult topics (concepts) according to instructors and students from the reviewed literature. It suggests that Recursion, Pointers, Functions, Abstract Data Types, and Arrays are the CS1 topics considered most difficult. The literature also shows that other than the difficult topics, algorithm design and problem solving [44, 124, 128, 211]; syntax [124, 128]; and debugging [124, 156] are common issues that make programming difficult. Others are lack of student motivation, lack of maturity, study skills, student background and poor mental models.

Although several topics/ concepts and issues have been identified as difficult in programming, in this thesis, recursion is used as a case study as explained in Section 1.5. A tool is proposed to support programming instructors to generate games to teach the concept of recursion in CS1.

2.3 Recursion Difficulty

Previous studies have pointed out that the recursion topic is difficult for many students in higher education [172, 146, 233]. The concept is seen as obscure, difficult and mystical for most students [219]. While investigating the topic difficulty, one study found out that unfamiliarity with recursive activities, difficulty with visualizing program execution, and difficulty in understanding recursive flow of control are some of the reasons why students struggle with the concept [105]. Additionally, in a review of over 35 publications on the topic, lack of backward reasoning, misconceptions about base case and inability to form viable mental models were identified as some of the difficulties that learners face while learning the topic [160]. Other reasons for the recursion topic difficulty have been identified as: lack of preparation in logic and maths skills; miscommunication between instructors and students; poorly designed instructional books and topic coverage; poorly designed projects and code examples; ill prepared instructors; and inadequate localization/ contextualisation [48].

Some works have however argued that the approach taken by the instructor may either make it easy to learn the recursion topic or not [84, 133, 175]. On the other hand, while teaching recursion in procedural programming, one study observed that teachers should first emphasize the declarative and abstract level of divide-and-conquer and also take precaution to strongly relate to the basic computing model [84]. Mirolo investigated learning recursion in a multidimensional perspective and suggested that a

comprehensive model of recursive computation was fundamentally linked to the skills that allowed learners to apply recursion in problem-solving tasks [175]. While working with 8th graders, Lonati et al. [133] showed that while relying on self-similarity and delegation, recursion strategies might be within the proximal development of pupils.

Another study found out that effective strategies for introducing recursion could include; (i) a description of how recursive methods are processed using a call stack; (ii) teaching loops before recursion; (iii) using well designed games; (iv) using different contexts like recurrence relations; and (v) using programming examples such as fractal images [160]. The Approach Deployment Result Improvement (ADRI) model was proposed to enable lecturers to place more emphasis on problem solving strategies [105]. Additionally, some researchers proposed the use of serious games to demystify and illustrate the concept [48, 255]. Although, several methods have been advanced to teach the difficult topic of recursion, a recent study found that there is little evidence to demonstrate the effectiveness of most of them [142].

Moreover, despite these prior strategies, recursion - a topic traditionally considered difficult - still remains universally so for novices in developing and middle income countries as is the case in other parts of the world [17]. Part of the difficulty is because CS1 attracts a large and very diverse audience [176, 230, 257]. This is further amplified by the fact that the course is studied not only by CS majors but also students from other disciplines [230, 53]. This leads to a class of students with mixed abilities and different motivational needs. As such, the use of serious games to aid teaching difficult topics such as recursion in the course has been an area of interest to some scholars [48, 58, 130, 255] in an effort to find new and fun ways to engage the diverse net generation students [193] with learning. Section 2.4 presents a global overview of literature on some of these GBL efforts and serious games design and development.

2.4 Game Based Learning Efforts and Conceptual Game Design

This thesis is about a game generator tool designed to support CS educators to easily create games to teach programming using the recursion topic as a case study. In order to achieve this goal it is important to survey the landscape of existing programming educational games and their design features. This potentially inspires the development of a prototype of the tool under consideration. It offers an opportunity to identify design gaps and understand the desirable programming game design features. This section provides a review of serious games specifically designed to teach the recursion topic in CSE, some conceptual design considerations and the general serious games development process.

2.4.1 GBL efforts

Given the difficulty of recursion, GBL has been proposed as one of the possible interventions by some computer science education community researchers [58, 255, 209]. This proposal makes even more sense for the sub-Saharan African student who, due to the lack of infrastructure and low household income, has inadequate exposure to technology during formative years of schooling [67, 182]. Games have the potential to increase the motivation of students by keeping them attentive and immersed [81]. Indeed, as is supported by a body of evidence, other reported benefits include self directed learning [247], scaffolding [79, 209], short feedback cycles [60], and strategic thinking[224]. Wilson et al. report that serious games can improve some aspects of learner performance such as knowledge retention, problem-solving skills, communication, motivation and attitude change [274]. Additional benefits include stimulation, collaboration and adaptability [108]. Acknowledging these advantages, the authors of a recent study encouraged CS teachers to adopt GBL [118] in teaching. Moreover, another recent study with CS1 found that *"games (83%) and STEM (98%) are popular interests that students perceive as relevant to CS/ coding"* [45, p. 5].

Owing to such benefits, some worthy efforts have been made towards developing programming games to teach the concept of recursion. For example, Dann et al. attempted to teach the concept using visualization with Alice - a 3 dimensional interactive graphics programming environment [58]. Results suggested that visualization could be used to develop good intuition about recursion. In another study, Cargo-Bot was a video game on the Apple iPad where players controlled visual robots by creating simple programs using a visual language [255]. Findings from the controlled experiment showed a significant improvement in the ability of students to write recursive algorithms.

Additionally, the work by [255] was built on by Lee et al. [130] with a modification of the Cargo-Bot game to include a visualization of the program stack and adjusting the rating system to reward the cleanest and most elegant solutions. Findings showed an improvement in student learning of recursion over traditional lecture-based instruction alone [130]. Another study by Chaffin et al.was about Elemental - a video game that involved completing three programming puzzles [48]. The game demonstrated the potential of teaching students the concept of recursion by coding Depth First Search (DFS) for a binary tree then allowing them to visualize [48, 208] the execution.

Table 2.2 presents a comparative analysis of design features of five serious games specifically created to teach the recursion topic in CS1. They include Elemental [48], Cargo Bot [255], Light Bot [2], Program Your Robot [113] and Saving cera [26]. The inclusion criteria used is based on: (i) games data sourced from Association of Computing Machinery (ACM) Digital library, Google Scholar, Institute of Electrical and Electronics Engineers (IEEE) Xplore, Springer Link, Science Direct; (ii) games re-

leased between 2000 and 2018; (iii) educational games only; (iv) games for teaching recursion in introductory programming; and (v) online availability for play.

Elemental the Recurrence, is a game designed to teach students recursive algorithms [48]. Players write and compile their codes using C#. The first level is a simple 'Hello world' program to introduce students to C# programming. Students walk a binary search tree in a Depth First Search (DFS) pattern - a recursive algorithm. In Level 2 of the game, learners complete a half way competed DFS algorithm where they code the movement of a character inside the game [48]. Finally, they compile to visualise the character traverse the tree. Figure 2.1 is an illustration.

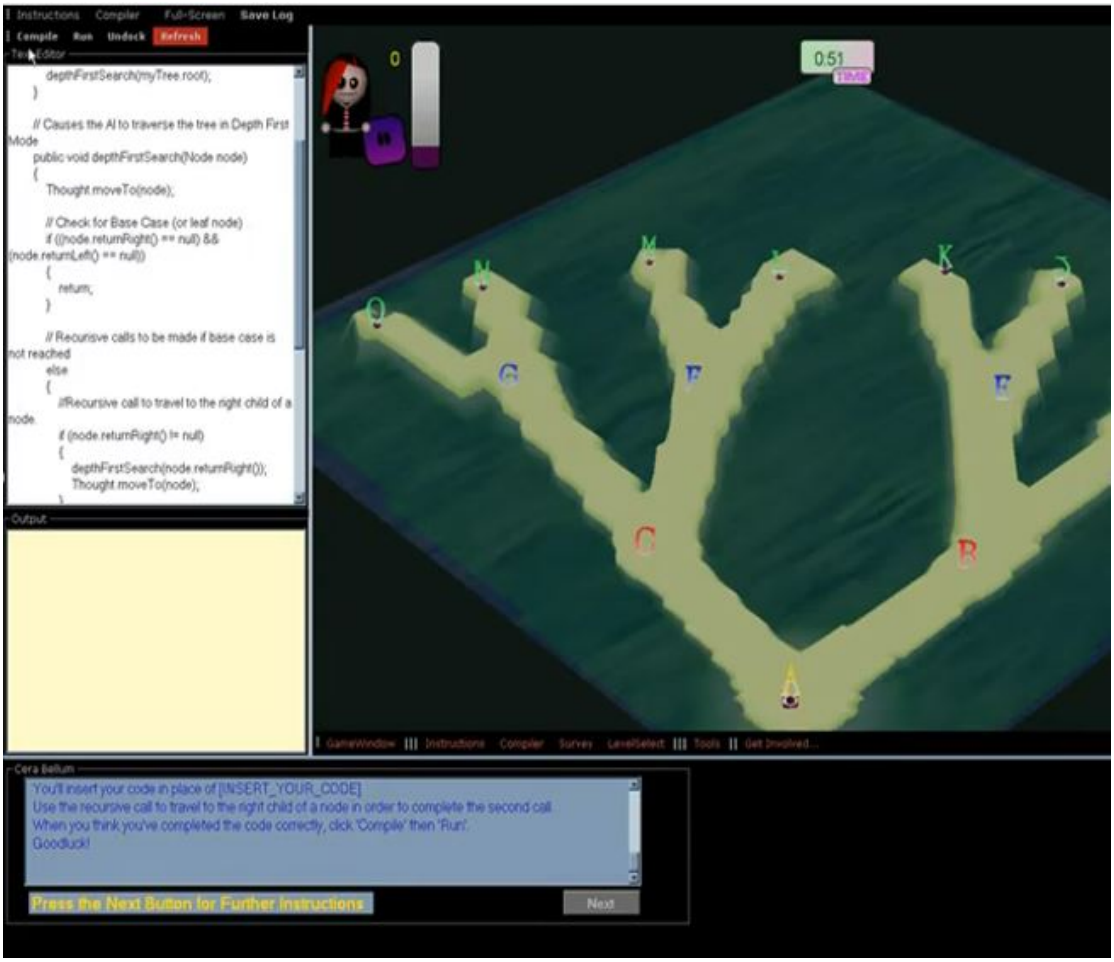


Fig. 2.1 The Elemental Recurrence Game - Level 2 [48]

Cargo-Bot game is programmed on iPad to teach basic programming constructs. To play the game, a user programs a claw that grabs cranes and moves them [255]. The movement is controlled by the commands selected by the player. These commands are available in a toolbox. Thereafter, the chosen commands are dropped in the drop zones labeled program 1, 2, 3 etc. Once this is done, a player clicks the play button to visualise the execution of the algorithm by how the craw grabs the cranes and moves them. Figure 2.2 is an illustration of a game play scenario.

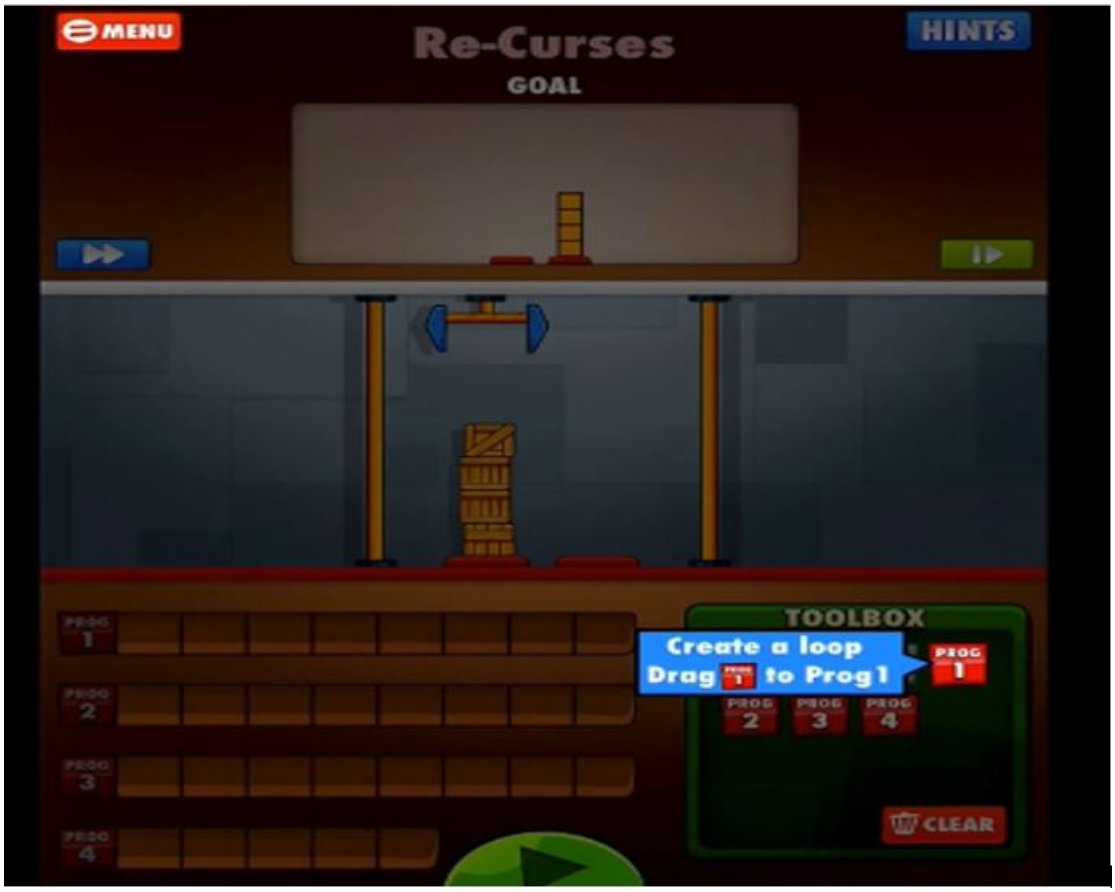


Fig. 2.2 Cargo-Bot Game Play Scenario [255]

Light bot is a game that teaches programming through solving puzzles. Players can learn sequencing, overloading, procedures, recursive loops, and conditionals while playing the game. The game can be played on an Android mobile device. A player programs a robot to light up boxes by dragging and dropping commands [2]. Some of the available commands include the right, left, jump, light, step, and break, among others. Speakers are also designed to allow players to leverage on the sound effect. Figure 2.3 is an illustration of one game play scenario.

From the analysis of the programming games reviewed in this Section, we learnt that that Alice, Cargo-Bot and Elemental were successfully in teaching students the concept of recursion because of the visualisation feature. Meanwhile, Elemental and Saving Cera games worked in helping students to learn programming by construction due to the built in Integrated Development Environment (IDE) that allowed coding and debugging. Cargo-Bot and program your robot games were successful in teaching programming as they had clear defined learning goals and they offered scaffolding. On the other hand, it seems the five games did not work because they did not allow for customisation, use of simple examples and did not consider culture. These lessons were further investigated

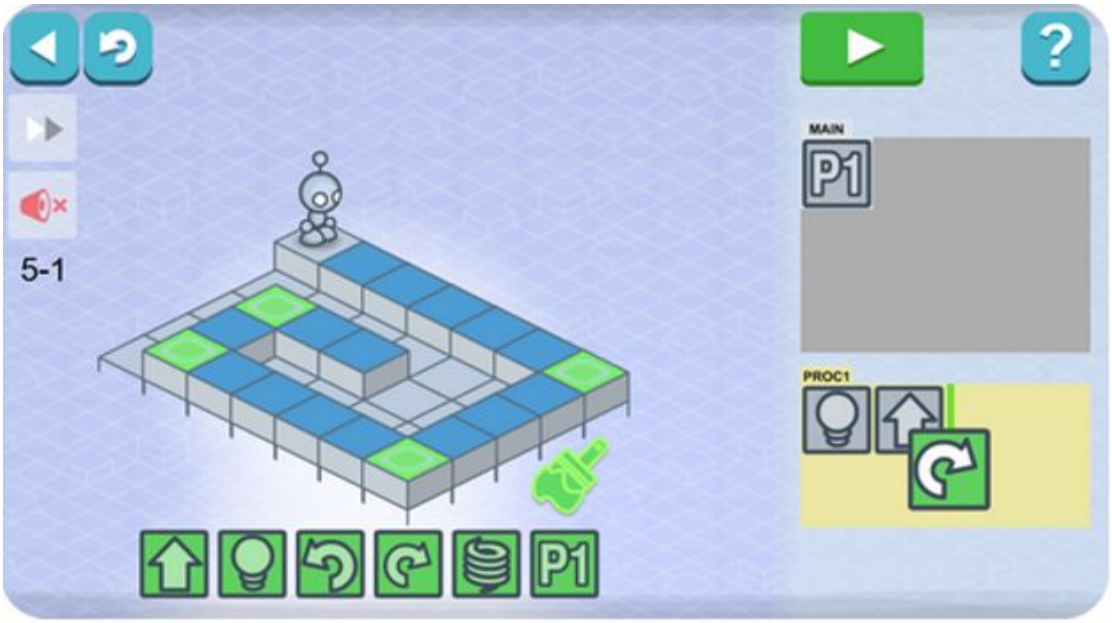


Fig. 2.3 LightBot Game Play Scenario [2]

in Sections 3.3 and 3.4 when understanding user requirements and applied in Section 4.6 (prototype design and development).

2.4.2 Conceptual game design

2.4.2.1 Educational theories and game design

While reviewing literature on theories about game design principles, Gunter et al. [1] defined game design as the formal methods for specification and planning of content and features for video games. It is important to balance pedagogy with the entertainment intentions when designing serious games. However, the challenge lies on mapping core game mechanics with specific instructional objectives. Constructivist and behaviorist pedagogical theories [61] could help to understand this mapping.

Proponents of constructivist theory argue that knowledge is actively constructed by learners who are facilitated by teachers to develop patterns, raise questions, solve problems and reflect on their ideas [75]. Students learn through active involvement, experiencing and exploring the world around them [121]. Piaget believes that children develop cognitive structure through action and spontaneous activity [200]. In his work, Papert further refines the constructivist theory by adding involvement of learners in the actual design, construction and creation of artifacts [196]. In order to develop the capacity to learn, Piaget's constructivist approach to learning is anchored on: (i) stimulating interest; (ii) initiative; (iii) experimentation; (iv) discovery; (v) play; and (vi) imagination [200]. Vygotsky contends that learning takes place through social interaction between the children and adults [270]. This could be extended to the learning that takes

place between experienced and less experienced learners. Piaget and Vygotsky appear to share 3 key principles of constructivism [158], namely:

(i) Knowledge is gained when a learner undergoes an experience that enables them to learn.

(ii) Learners are active builders of their own knowledge through expression and interacting with other people or things in their environment.

(iii) A relation exists between existing knowledge and any new knowledge acquired by the learner.

Behaviorists on the other hand argue that learning is a system of behavioral responses to stimuli. In this theory, learning takes place through association, practice, external motivation and by reinforcing the learned behavior. Learning tasks/ activities are broken down into sub tasks/ activities and sequenced in a structured order according to Bloom's taxonomy [99]. The tasks are then given to learners to solve. This is followed by evaluation and reward at each stage, giving learners an opportunity to learn concepts gradually.

Conceptually, regarding cognitive theory, serious programming games should have environments where players can gain knowledge through exploration and practice [61]. Meanwhile, Cognitive Load Theory (CLT) underscores the need for explicit teaching and guidance given the limited working memory [266]. The conceptual design implication is that the game generator tool should create games capable of reinforcing programming concepts already taught in class. The game activities should be designed at the level of players with clear guidance offered, especially, to the novice learners. Additionally, game elements such as maps, contextualised help, and objectives'lists should be designed to avoid players'cognitive overload [61]. Table 2.3 is a summary of the constructivism and behaviorism theories and the conceptual design implications.

2.4.2.2 Conceptual design attributes/ features/ aspects

Some early works on educational games design include those by Harteveld et al., [93], Malliarakis et al., [148], McGill et al.,[163] and Merero et al., [167]. For instance, some of them underscored the importance of thematic conceptualization of educational games based on a number of aspects to determine what features should be supported when specifying requirements during design [93, 148]. In particular, Malliarakis et al. identified features that ought to be considered when designing and developing an educational game to teach programming [148]. These include: (i) motivating scenarios; (ii) clearly defined educational goals; (iii) clear problems for students to solve; (iv) facilitating tools such as a programming editor; (v) information resource (explanation facility); and (vi) one or many frameworks [148].

Additionally, this study summarised four conceptual design features supported by most commonly known educational games for programming as: (i) programming activities - filling lines of code, answering multiple choice questions, completing missions, dragging and dropping lines of code, moving characters and writing full code in an editor; (ii) programming elements - the programming concept(s) to be taught; (iii) programming language to be used; and (iv) special characteristics - two/ three dimensional, online availability and scaffolding [148]. Features such as clear educational goals, problems to be solved, and programming editor goals are aligned with the constructivism theory. Laporte and Zaman argued that considering novices'programming problems such as learning content when designing programming games could yield games that suit their needs [128].

In a consolidated evidence-based review on the design and use of serious games in higher education, Lameras et al. [126] identified two key conceptual attributes worth considering during design. These are learning and game attributes. Learning attributes include learning activities, learning outcome, assessment, feedback, and role of the teacher. Game attributes entail levels, game hints, scores, game narration, extrinsic and intrinsic rewards, avatars, dialogues, rules, challenges, collaboration and competition.

In another study, educational game elements such as narrative context, rules, goals, rewards, multi-sensory cues, and interactivity were seen as necessary design aspects to stimulate desired learning outcomes [68]. The learning attributes, game attributes and the educational game elements are well aligned with the constructivist and behaviorist learning theories. Some works have recommended that game developers, art designers and domain experts (teachers) should work together [170, 177]. However, this is not practically easy [263].

2.4.2.3 Conceptual models

Meanwhile, some scholars have proposed conceptual models [258, 281]. For example, when designing educational games for effective learning, Yusoff et al. proposed a conceptual model comprising of: (i) capability; (ii) instructional content; (iii) intended learning outcomes; (iv) game attributes; (v) learning activity; (vi) reflection; (vii) game genre; and (viii) game mechanics [281].

Recently, Canteri et al. [70] proposed the Conceptual Framework for Educational Games Design (CFW) that uses modularisation approach. In this model, four modules support the creation of serious games for the deaf children. They include Game Play and Tutoring, Teaching-Learning, Student/ Player, and Graphics and Interface module [70]. The Game Play and Tutoring module supports the design of game mechanics and aligns them with the content the teacher wants to teach. The Teaching-Learning module encompasses the educational content to be learned and the order in which this

should take place. The student/ player module gives the learner an opportunity to get feedback on the success or failure when performing the designed learning activities. It also adapts the learner's game play experience to different difficulty levels. Lastly, the Graphics and the Interface module takes care of issues such as the student interface and interaction environment, artistic style, scenario, character appearance and color [70]. Figure 2.4 shows the Conceptual Framework for Educational Games Design.

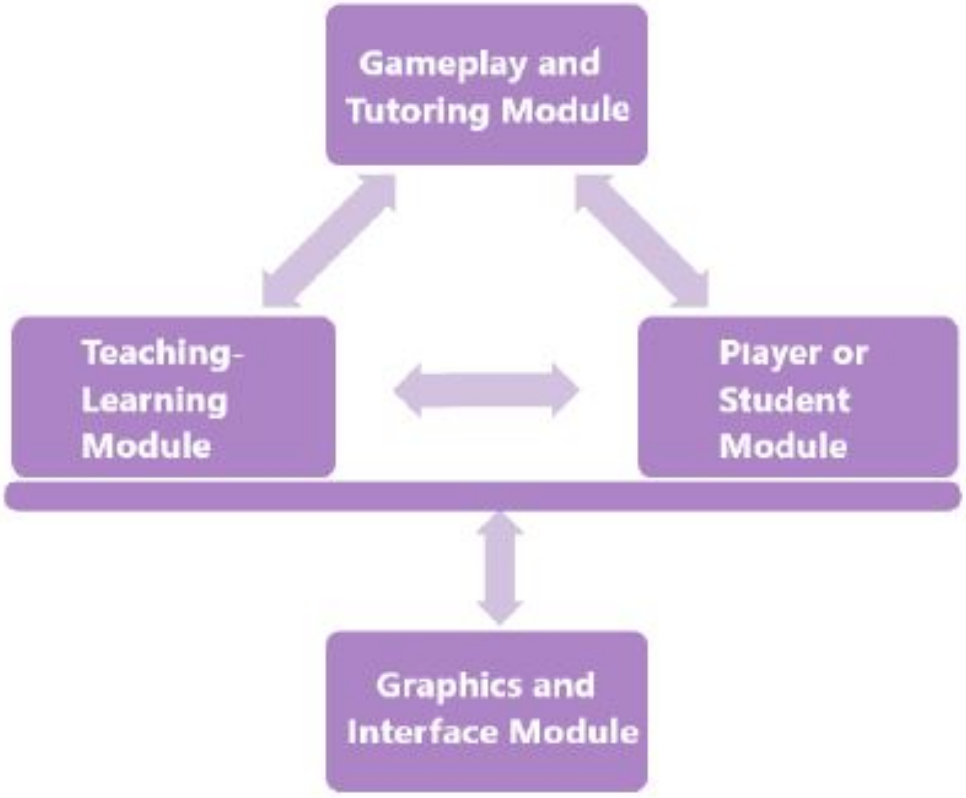


Fig. 2.4 Conceptual Framework for Educational Games Design [70]

On the other hand, in another recent study, Tondello et al. [258] explored personalised game design and proposed a trait model and a measurement scale for game play preferences. In this model, five player traits are proposed as: (i) social orientation - the extent to which learners/ players are oriented to playing online with others or in the same space; (ii) aesthetic orientation - player preference for aesthetic experience such as exploring the game world, scenes etc.; (iii) action orientation - player preference for challenge; (iv) goal orientation - player preference for competently completing game quests, tasks etc.; and (v) immersion orientation - player preference for complex game stories on narratives [258].

The Learning Mechanics - Game Mechanics (LG-GM) model not only combines learning and game mechanics in the design process of a serious game but also analyses and assesses the game. The learning mechanics include motivation, exploration, ob-

servation, reflection, analysis, feedback, participation among others. Meanwhile, some of the game mechanics are role play, collaboration, tokens, movements, action points, feedback, game story, and scenes. The model *"provides a method to validate the suitability of game mechanics in regards to how they match with different pedagogical approaches"* [4, p. 1].

The Activity Theory Model for Serious Games (ATMSG) is useful during conceptual game design since it enables game designers to know early enough at the prototype stage whether or not a game structure design will support the envisaged pedagogy [47]. According to ATMSG, educational serious games are used in light of four activities: (i) the gaming activity; (ii) the learning activity; (iii) the intrinsic instructional activity; and (iv) the extrinsic instructional activity [47]. This model is anchored on the activity theory with the emphasis that serious games are not considered in isolation but in the context of a complex system comprising human actors - students, teachers, designers etc. Also considered is what motivates them to interact with the games. The model maintains that the leaner can use some educational games by herself/ himself but some games may require instructional activities and help of an instructor. Consequently, the model advocates for intrinsic and extrinsic instruction activities during game design [47]. Whereas intrinsic instruction activity is inside the game i.e. how the game supports learning using tips, scaffolding, automatic assessments etc., extrinsic instruction activity is performed prior to and after game play session such as organising a class session [47].

Another model is the Relevance Embedding Transfer Adaptation Immersive Naturalisation (RETAIN) [88]. It was developed after the authors identified significant elements between game and instructional design that were either missing or misunderstood by game designers and educators. The model was intended *"to create games that would foster learning, retaining, transferring, and naturalizing academic content"* [88, p. 9]. It is anchored on three foundational instructional design theories: (i) Keller's ARCS motivational model [114, 115]; (ii) Gagne's events of instruction [77]; and (iii) Blooms taxonomy [99, p. 9]. In order to design instructional activities, Keller's model depends on four aspects i.e. Attention, Relevance, Confidence/ Challenge and Satisfaction/ Success [114, 115]. Meanwhile, Gagne proposes nine events of instruction. Blooms taxonomy ensures game design at different levels of complexity. The RETAIN model combines elements of the three theoretical foundations into six game design tenets [88], namely:

(i) *Relevance*: where learning materials should be provided in away relevant to learners, their needs and learning styles. Thus, the material should allow learners to build on already acquired knowledge.

(ii) *Embedding*: where learning content should be carefully mapped with game fantasy and immersion.

(iii) *Transfer*: the design element that ensures that learners/ players can transfer and apply knowledge to new similar situation or problems.

(iv) *Adaptation*: the design element that ensures that learners can recognise patterns and transfer current knowledge to similar new circumstances.

(v) *Immersion*: where game design should ensure that players are fully immersed during game play. This can happen at different levels.

(vi) *Naturalisation*: where a learner can apply the learnt knowledge spontaneously with minimal cognitive load. Regarding game design, it means that the game is engaging and motivating to the extent that players/ learners are willing to play it over and over until it becomes automatic how to play it.

In a recent comprehensive literature review, Ahmad et al. [4] reviewed the: (i) LG-GM; (ii) Activity Theory-based Model of Serious ATMSG; and (iii) RETAIN serious game design models. The authors castigated them - alluding that they required knowledge of game design. This makes it difficult for educators to fully utilise them on their own without involving game designers [4]. Meanwhile, the researchers who proposed the RETAIN model also recognised the fact that not all the elements are important and that further studies are needed to establish the elements that need to be weighted more or less [88]. Table 2.4 presents a detailed summary of common serious games pedagogical conceptual design considerations while Table 2.5 shows those of game play design consideration as adapted from ([68, 101, 126, 148]).

2.4.2.4 Programming games design

Existing educational games specifically designed with a focus on computer programming do not enable teachers to configure the game environment according to the pedagogical goals related to the respective unit of learning [148]. In line with this finding, another study reported lack of tools, scaling and personalized teaching as the challenges CS1 teachers face [164]. Meanwhile, the work by Vahldick et al. [265] searched a sample of games designed and published on the Web after 2000 to improve introductory computer competencies and presented a summary of 40. Out of these, only 10 taught the concept of recursion, among other topics. Moreover, only two could allow more than one language. Additionally, none could adapt their interaction according to evolution of students [265].

Within the CS research community, countless initiatives have been geared towards diversity, equity, and inclusion in CS curriculum [45, 51, 101, 132, 178, 188]. The

hope is to address the shrinking pipeline in STEM [54] and bring on board the under represented minority groups. To bridge this gap, GBL has been tried. However, conceptually, designing serious programming games that address diversity remains a challenge. For example, a recent study showed that White, Asian, Hispanic / Latin, and African American / Black students selected "A lot" 51.83, 48.41, 27.20, and 38.80% respectively when asked if they had an interest in video games [101]. The tool proposed in this study attempts to create games that consider diversity but in the context of Africa. Nonetheless, consideration is given to universality in design for world wide testing.

Although previous studies provide some useful insights on serious games design, there is a lack of sufficient information and guidelines on how to design and develop games to teach computing competencies that consider both game play and pedagogy [29, 108, 163]. Furthermore, most material describing games for teaching computing in higher education does not explain how these games have been designed and developed [29]. Additionally, there appears to be no attention given to diversity in design. This is illustrated in a report by a special working group on game development for CSE noting significant design challenges and opportunities offered by diverse learners as a result of different gaming backgrounds [108].

While the early initiatives towards teaching the recursion topic through GBL approaches are commendable, developing playable educational games still remains difficult, time consuming and expensive [87, 260]. This hinders domain experts (educators) with little game programming skills who may wish to use GBL approaches, leading to low GBL adoption in mainstream teaching [252]. The adoption rate is particularly low in the context of Africa. Consequently, additional work is desirable if the potential of GBL is to be fully realised. Section 2.5 offers a review of the barriers to GBL in mainstream teaching.

2.5 Barriers to GBL Adoption

There has been growing interest in GBL in CSE education [249, 30, 283]. This trend is being driven by the reported educational value of serious games [155]. Nonetheless, the adoption rate of GBL in classrooms by teachers remains low [4, 261]. According to Torrente et al.[261], barriers to GBL adoption in teaching can be attributed to three broad issues: (i) design stage issues (harmonising educational value and fun); (ii) production stage issues (cost); and (iii) deployment stage issues (delivery, distribution and classroom implementation). It is difficult to design games with educational value that are aligned with the curriculum [228] and also entertaining at the same time. Moreover, the production costs are high. The situation is further worsened by ill prepared teachers and insufficient time in the curriculum [261].

Battistella and Wangenheim [29] also point out the practical challenge of introducing games in class considering the planned teaching time in the CS1 curriculum. Meanwhile, absence of appropriate game authoring tools and support for non-technical domain experts such as educators has also been seen as slowing down many who wish to adopt a GBL approach [111, 253]. In fact, results from a survey of 30 instructors showed that there exists a positive relationship between teacher's experience in designing games and the actual usage of games for educational purposes [7].

Echoing the same concerns, Bontchev and Panayotova describe three technological barriers to GBL adoption [37]. These include: high cost of games; difficulty in finding games relevant to curriculum and difficulty in integrating game and curriculum contents. Another study notes that: (i) failure of technology; (ii) the fact that not everybody likes games; (iii) the enormous amount of preparation required by teachers within a limited time in CS1 curriculum; and (iv) the fact that most games are not serious are the other potential hurdles to games use in introductory computer science courses [30]. On the other hand, large amounts of development time and expert knowledge required from developers and educators have also been identified as potential barriers to the adoption of GBL in teaching [14, 179]. Baek identified six barriers that hinder the adoption of computer and video games by teachers [22]. These include the: (i) inflexibility of the curriculum; (ii) perceived negative effects of gaming; (iii) unprepared students; (iv) lack of supportive materials; (v) fixed class schedules; and (vi) limited budgets [22]. Recently, Ahmad et al. reported that creating customised serious games is difficult, expensive, and time consuming as it involves instructional designers (domain experts), game designers and game developers working together [4]. This cooperation is necessary since the domain experts can easily design pedagogical instructions but cannot code, while game designers and programmers can design and code but cannot design instructional content [145].

This thesis attempts to address some of the identified barriers by proposing the idea of a teacher-oriented game authoring tool capable of enabling any teacher with basic computer skills to easily create and customize their own educational games [134]. Some prior works have reported that authoring tools reduce development time, effort, and cost as well as lower the skill barrier and allow customisation [6, 91, 169]. The central argument is that supporting CS educators with a game authoring tool could potentially make them more effective and efficient when creating educational games for teaching CS1. Specifically, the goal of the current study is to: (i) reduce serious games production time, cost and effort; (ii) increase integration of fun and educational value in serious games design; (iii) allow for customisation in serious game design; and (iv) reduce deployment stage issues. The thesis extends the existing body of work in GBL, but differently, by adopting a teacher centric approach as in [206]. Next, Section 2.6 presents game generators.

2.6 Game Generators

A game generator is a software platform dedicated for non-experts that allows users to generate video games according to their needs/ specifications [72]. In this work, literature on game generators is discussed under four subtopics: (i) procedural content generation; (ii) commercial game generators; (iii) serious game generators and; (iv) empirical evaluation.

2.6.1 Procedural content generation

Within the game design and creation space, only few researchers have considered automatic game generation. Most previous works on automatic game design were predominantly about procedural content generation [74, 87, 153]. Whereas Dong and Barnes [69] as well as Fernandez-Vara and Thomson [74] investigated procedural puzzle generators, Green et al. [87] and Khalifa et al. [116] looked at level generators. Mark et al. [153] investigated procedural generation of 3D caves for games. These works employed procedural content generation as a way of saving time and automating a set creation.

While describing a flexible approach to game rule-set generation, the study by Smith and Mateas [241] considered both automatic game authoring and procedural content generation. In the commercial area, Gaming 2.0 tools were created to ease the design process of video games [65]. However, such tools have limitations when it comes to creating mechanics, story or interfaces relevant to entertainment games. Moreover, the tools are not suitable for designing serious games.

2.6.2 Commercial game generators

Some well-known low cost game authoring tools in the market include Creator, Game Maker and Mission Maker. However, they are designed for commercial purposes. Consequently, they lack pedagogical design aspects hence are not suitable for creating educational games. The study by Treanor et al.[264] presents another commercial game generator tool caled Game-O-Matic. This is a simple authoring tool capable of automatically generating simple arcade games in the genre of newspapers using a rule-based approach based on input that lists objects, actors and their relationships. However, basic rules are combined into playable games, which might present difficulty to non game experts as well as non programmers. Moreover, the generator has never been formally evaluated.

2.6.3 Serious game generators

Some researchers have attempted to design serious game generators. For instance, e-Adventure[1] is an instructor oriented game authoring tool for educational point-and-click video games [260, 262]. The tool supports instructors with no programming background to adapt games already created by others or create their own from scratch. The generated games are integrated in e-learning platforms as learning objects. Additionally, the platform has a built-in learner assessment mechanism. The editor window allows instructors to edit elements of a game such as objects, characters, books, and scenes.

As an improvement on eAdventure, another serious game authoring tool called uAdventure was built on top of the Unity game engine [198]. The authors argued that the eAdventure editor had technologically become obsolete since its deployment relied on Java Applets, which are no longer supported by most browsers. Further, they claimed that the authoring tool could only support desktop platforms. Given these limitations, uAdventure was designed to: (i) address the technical issues; (ii) face the survival issues; (iii) provide a solid base platform to build new educational features; and (iv) ensure compatibility with eAdventure's past games [198].

Another work that attempted to improve the serious game authoring process was the Mokap toolset project [263]. It aimed at two things: (i) making authoring tools easy to use by the educational community; and (ii) supporting design and analysis stages . The tool's efficacy to support agile development of serious games was demonstrated through the total number of hours (130) and lines of code (9000) used to convert specifications given by domain experts into the Listen with Nemur game for the hard to hear children aged between 18 - 24 months old. The authors claimed that the figures were quite low compared to the industry standards [263].

Eaachak et al. proposed a serious game generator dedicated to non-experts such as educators to make cross platform serious games [72]. Evaluation results of one of the generated game prototypes showed that the study participants (five children) enjoyed playing the game. Regarding the pedagogical objective - teaching children how to use a computer mouse, there was a remarkable improvement on speed (26.77% faster than the first attempts)[72].

A semi automatic generator of tactile video games to help visually impaired children to develop their touch, familiarize themselves with Braille terminals and to help them in Braille learning was proposed by Sepchat et al. [235]. A few tests were done with visually impaired adults who were advanced Braille readers and people without any visual impairment to validate the generated games. Results showed that participants without visual impairments quickly understood the games, but not those with.

[1] http://e-adventure.e-ucm.es/ (Accessed May 25, 2020)

Bouzid et al. [41] suggested a game generator tool capable of generating several instances of MemoSign game according to user input data and the chosen target platform as an educational technology for deaf learners. As currently designed, it can not be adapted to create games to teach programming. Furthermore, no experimental evaluation was done to evaluate the effectiveness of the tool on creating the MemoSign game.

Another effort in automatic game generation is the Extensible Graphical Game Generator (EGG) - an experiment in automated programming [194]. In this experiment the similarities in poker, chess, tic-tac-toe, rock-paper-scissors and many other classic game programs are adapted to create a universal game engine called EGG. This generator enables users (game designers with little programming knowledge) to automatically create playable games by simply describing the rules that differentiate their game from more generic examples of the genre to the engine. However, in order to generate a new game, the author must describe the game in terms of the rules of play. As with Game-O-Matic, domain expert authors may find it difficult to specify the rules. This might prove challenging for programming instructors with little inclination towards game design and development.

The study by Landro [127] described a data driven game generator where the logic resides in data rather than in the system. Users (educators) generate different instances of games by changing the input data. The generated games consist of assignments, each being a task to be solved by a player in order to learn system development concepts. Handlers are used to determine the next assignment based on a players response to the current assignment. A custom handler allows authors to make their own handlers facilitating extension of the system without changing code. Though the authors performed a qualitative test on the system to evaluate its usability and user friendliness, the design does not consider conceptual and material elements of game design.

In the health domain, Simulate Coach Review (SimCore) is a framework with an authoring tool for rapidly creating and editing game scenarios comprising: locations, props, and events [244]. The tool supports course developers and health care trainers to rapidly come up with and modify training scenarios through visual editors. The tool was developed in three spiral cycles with a formative evaluation at the end of each with health experts. Though an attempt was made to evaluate the tool, it was mostly formative. A summative evaluation would have given more insight on its usability. Within the programming domain, Khenissi et al. implemented the Learning version of Pacmac Game Generator (LPG) and the Instruction Right Place Game Generator (IRPG) to enable teachers to create games in 4 and 5 simple steps respectively [117]. For instance, using IRPG, a teacher could create an instance of the IRP game using the following simple steps:

(i) Step 1: Add game name and assign level.

(ii) Step 2: Write the questions.

(iii) Step 3: Write solution to the questions.

(iv) Step 4: Add an evaluation question.

(v) Step 5: Click save to create game.

Although IRPG uses simple steps, the generator is designed to solve problems in the Maple programming language. Similarly the LPG tool teaches the Maple programming language. The generator in this thesis seeks to deviate from these two by aiding teachers to easily create games to teach novices programming by practicing Python programming - one of the most popular language for teaching introducing programming to novices[33, 131, 213, 223, 238].

Machiori et al. present the Writing Environment for Educational Video Games (WEEV) system - an instructor oriented game authoring tool [150]. The tool is implemented using three separate tools: an actor editor, story editor and a world editor. Meanwhile, Torrente et al. observed that one major limitation of current game authoring environments is the fact that they do not support game design processes from the domain experts'(educators') point of view [263]. They argue that this limits the participation of this category of important stakeholders and hinders adequate communication of user requirements from a pedagogical point of view as well as support for agile/ rapid prototyping. The design and development of the game generator tool in this thesis puts programming instructors at the center stage as key informants.

Yaelle et al. [49] developed an assessment engine called EngAge for serious game developers and educators. The tool's visual editor allows teachers to create and modify games based on the needs of their students. The aim of the study was to address three reasons why teachers do not trust serious games assessment - namely: the feeling of not being in control, lack of games ownership and lack of detailed reports on game play to demonstrate student learning [49]. Empirical evaluation with teachers and developers demonstrated the usability of the tool. The main focus of this work was a game assessment engine. The current work focuses on a game generator platform capable of creating games to teach programming.

2.6.4 Empirical evaluation

In order to provide reasonable evidence to show that an educator could effectively use a game generator (authoring) tool to create an educational game, there is need for empirical evaluation. Some researchers take the view that empirical evaluation of a game generator tool ought to be pegged on finding out how well the generated games score against the generation criteria [13, 55, 151, 195]. Such criteria are the targets set by

the tool designers such as game: (i) quality; (ii) inventiveness; (iii) video/ audio excellence (iv) length (v) constraints on solvability; (vi) mechanics; or (vii) any other design parameters. They claim that human judges through user experience evaluation could measure the success or otherwise of a game authoring tool. Some authors have identified usability evaluation criteria for game authoring tools for teachers as: (i) easy to learn; (ii) error prone; (iii) efficacy; and (iv) efficiency [24]. Some empirical studies and their results follow.

Study 1

Osborn et al. conducted pilot studies using questionnaires and a controlled experiment with 76 students to see whether a generator called Gemini could create interpretable games [195].

(i) Result 1: Participants agreed with the tool about lower level mechanics present in games.

(ii) Result 2: Participants successfully identified mechanics of two games.

(iii) Result 3: Participants did not extract high level meaning of games given abstract design.

(iv) Result 4: No statistical relationship was seen between how 2 groups of players understood mechanics and how well they could interpret meaning from it.

Study 2

Khenissi et al. conducted an experiment with 167 first grade students of a tertiary school in Tunisia to evaluate the effectiveness of serious games created using IRPG and LPG generators compared to learning versions of existing games [117].

(i) Result 1: Suggested a positive impact on students'knowledge gain.

(ii) Result 2: Answering a questionnaire on usefulness, ease of use, intention to use, and attitude towards use metrics, students found generated serious games satisfactory.

Study 3

Anderson et al. conducted a controlled experiment with 43 undergraduate students to find out the psychological reality of procedural rhetoric [13]. They investigated whether or not the participants through their lived game play experiences could be able to interpret expert's rhetoric game design intents.

(i) Result 1: Players understood that two games intended to have arguments had them, except the abstract game (control) and that procedural rhetoric had psychological reality.

(ii) Result 2: Even if participants understood the goal of a procedural rhetoric argument in game creation, impact on their thoughts may not be the same as author intended.

Study 4

Eaachak et al. conducted a simple study with five children to evaluate the educational and entertainment value of a game generator tool [72]. The educational aim of the first game created was to teach children mouse skills.

(i) Result 1: Improvement on the mouse skills of the children.

(ii) Result 2: The children enjoyed playing the game.

Study 5

Gaeta et al. [76] carried out a usability study on an authoring tool for creating Storytelling Complex Learning Objects (SCLOs) in the ALICE project. The tool is made up of two components; an editor and a player and uses Microsoft Foundation Framework v3.5. Usability result was found to be relatively low. A System Usability Score (SUS) of 60.625 close to the satisfactory threshold of 68 was reported with minimum score 45 and maximum 77.5.

Study 6

Marchiori et al. [150] evaluated the WEEV system - a game authoring platform built on e-Adventure. Three evaluations were conducted: (i) formative (ii) end-user evaluation and (iii) game creation. The first two evaluations did not involve creating educational games. Instead, they assessed the impression and perception of educators.

(i) Result 1 - formative evaluation: 20 students found some information in the system excessive. Additionally, they appeared not to value the usefulness of the software.

(ii) Result 2 - end user evaluation: the tool improved the game creation process of nine educators who showed interest in using the tool. They also found the ability to test games as they are being developed useful but noted that the tool was complex to use.

(iii) Result 3 - tool testing by an English educator provided useful feedback from both pedagogy and usability standpoints.

Study 7

Tornero et al. developed e-Training DS - an authoring tool for integrating portable computer science games in e-learning [259]. The tool allows instructors to maintain a library of mini games that can be easily created by customising generic game patterns. In a case study, the time taken to create and modify the game and its assessment was measured. Results showed that the process was feasible for an educator. Results showed

that most tasks (65%) were rated as simple. For example, non-technical users (teachers) who had not interacted with the tool before found the story creation task much simpler. Meanwhile, there was positive feedback from more technical users while all users found the manual helpful. Nonetheless, participants identified 29 issues/difficulties with the tool for consideration.

Study 8

Perez-Colado et al. conducted the first preliminary user experience evaluation of the uAdventure authoring platform [210]. The tool was evaluated by heterogeneous users: (i) non-technical users - teachers (n=2), (ii) artists (n=2) and (iii) programmers (n=6) with different degrees of technical knowledge. The aim of the evaluation was to find out the issues/ difficulties with the tool. The difficulty of each performed task was measured on a scale of four levels: easy, normal, difficult or not-accomplished [210]. Another aim was to verify that users were able to:

(i) develop point and click game examples.

(ii) create non-linear stories with multiple endings.

(iii) automatically develop own stories.

(iv) build their games.

Study 9

Recently, another study performed a qualitative evaluation of zeusAR - a game generator tool designed to generate Augmented Reality Serious Games (ARSGs) [152]. The generated games were targeted at teaching geometry to secondary students. A usability evaluation was done with ten geometry teachers. The standard System Usability Scale (SUS) questionnaire was used. In addition, a performance analysis was done with 20 participants (the ten teachers and a team of facilitators drawn from different schools). The 20 who were both programmers and non-programmers used the generator tool to generate SGs [152]. Results of the usability questionnaire indicate that *"in most of the survey items the arithmetic mean is either equal to or higher than 3.5"* [152, p. 2926]. Further, the authors argued that the users had a favourable user experience and concluded that zeusAR was interactive and easy to use. Regarding performance analysis, results show that *"both expert users and non-expert users can generate ARSGs in relatively short period time regardless of the complexity of the generated games "* [152, p. 2928].

2.6.5 Analysis of the evaluation studies

Whereas the studies by Khenisi [117] and Eaachak et al. [72] evaluated the educational potential of the generator tools with students (the secondary users), those by Gaeta et

al. [76], Perez-Colado et al. [210], Marin-Vega et al. [152] and Tornero et al. [259] conducted UX and usability studies to demonstrate the effectiveness of the generator tools with teachers (the primary users). Marchiori et al. [150] evaluated both the educational potential with learners and the usability and UX with teachers. Meanwhile, only the studies by Osbon et al. [195] and Anderson et al. [13] were controlled. In 2 studies, only 5 participants took part-raising the question about generalisability of the results. Table 2.6 is a summary of the reviewed empirical evaluation studies of serious game generators.

2.7 Summary

This Chapter has presented a comprehensive review of related work. An overview of programming difficulty has been discussed in Section 2.2. The recursion topic difficulty was then presented in Section 2.3. Section 2.4.1 provided an overview of some GBL efforts while Section 2.4.2 highlighted conceptual game design. Section 2.5 explored barriers to GBL adoption in mainstream teaching. Lastly, Section 2.6 was on game generators with Section 2.6.2 specifically discussing commercial generators while 2.6.3 serious game generators. Section 2.6.4 was about empirical evaluation of serious game generators. Lastly Section 2.6.5 analysed the reviewed evaluation studies.

The reviewed literature suggests that programming is difficult for novices. As a way of dealing with the difficulty, GBL has been proposed. However, the adoption rate in mainstream teaching is low. Meanwhile, developing games is difficult, time consuming and expensive [111]. Lack of game authoring tools that can support instructors who wish to adopt GBL but have little game development skills has been identified as one of the adoption barriers [110].

For relevance to the broader CSE community, there is need to provide empirical evidence of the usefulness and effectiveness of using such tools by instructors. However, contrary to this expectation, the few studies that have attempted to demonstrate this are not in the programming domain.

Regarding empirical evaluation, Table 2.6 indicates that not much has been done to test the idea of serious game generators, particularly, to assess their usefulness, efficiency, and effectiveness through empirical user experience studies with educators. It appears that none of the current works on serious game generators has considered and evaluated a game generator tool capable of creating games to teach CS1 in higher education. This thesis hopes to fill this gap. Next, Chapter 3 presents a summary of 3 studies conducted to identify the problem and gather user requirements.

Table 2.1 Analysis of Difficult Programming Topics According to Instructors and Students

Study	Subjects	Topic (Instructors)	Topic (Students)
Bosse et al. [39]	Instructors (N=16) and students (N=110)	Functions, Arrays, and Loops.	Functions, Uni/ Multi-dimensional Arrays and Loops
Peteira and Costa [202]	Instructors (N=10) and students (N=64)	Pointers and references, Structured Data Types, Arrays,Error handling and Parameters, using language libraries	Pointers and references, Parameters, Structured Data Types, Abstract Data Types, Error handling
Lahtinen [124]	Instructors (N=34) and students (N=559)	Recursion, Pointers, References, Abstract Data Types, Error handling and using language libraries	Recursion, Pointers, References, Abstract Data Types,Function, Classes and Error handling
Milne and Rowe [173]	Instructors and students (N=66)	Pointers, Virtual functions and Dynamic Memory Allocation	Pointers, Virtual functions and Dynamic Memory Allocation
Schulte and Bennedsen [234]	Instructors (N=246)	Polymorphism, recursion, pointers and reference	
Dale [57]	Instructors (N=350)	Control structures, I/O and recursion and object orientation topics like inheritance, polymorphism, methods, classes, instance and static variables	

Table 2.2 Educational Games Designed to Teach Recursion (Elem= Elemental, CB= CargoBot, LB = LightBot, PYR = Program your Robot, SC = Saving Cera)

Game Feature	**Elem** [48]	**CB** [255]	**LB**[2]	**PYR** [113]	**SC** [26]
Supports writing code	✓	×	×	×	✓
Stack visualisation	✓	✓	×	×	×
Debugging feature	✓	×	×	✓	×
Drag and drop actions	×	✓	✓	✓	×
Goals aligned	×	✓	×	✓	×
Learning by construction	✓	×	×	×	×
Offers scaffolding	✓	×	×	×	✓
Freedom to explore level	×	✓	×	×	×
Allows customisation	×	×	×	×	×
Considers culture	×	×	×	×	×
Uses simple examples	×	×	×	×	×

Table 2.3 Constructivism and behaviourism pedagogical theories and conceptual Serious Games (SGs) design (Adopted from [99, 158, 277])

	Constructivism theory	**Behaviorism theory**
How does learning occur?	Emphasis is on creating meaning	When a proper response is demonstrated to a specific stimuli.
What factors influence learning	Learner and environmental factors	External stimuli.
How does prior learning affect new learning?	Engaging the learner in the actual use of tools in a real world situations	Through generalisation.
Game rules	Stresses the interaction among players and games, which are socially constructed.	Players learn the rules as mechanisms of the game in order to know what can be done and what cannot be done
Game play	Views learning as a social process and is not limited to the individual	Players know their goals and achieve these goals through stimuli-reaction process
Game narratives	Individual's perception of the game world is constructed by players interacting with each other.	Treat players as machines to be filled with information and they are expected to passively absorb the narratives.
Implications of theory on conceptual design	(i) Design should promote active learning through construction and exploration; (ii) the programming environment in the generated games should provide feedback mechanisms to the players when they make errors; (iii) the programming interface presented to novice learners should be as simple as possible; and (iv) pedagogical tasks at different game levels should be designed to support players to create new knowledge by building on the previous knowledge	(i) from the game play point of view, design clear buttons or commands to elicit specific reactions when invoked by the player. For example, clicking button A may result in a character moving forward while typing the command RIGHT() may mean moving right; (ii) design clear game reward systems through trophies, leader boards, badges, and points; (iii) for the game narrative, aspect design cut scenes or textual information from non player character.

Table 2.4 Summary of Pedagogical Conceptual Design Considerations

Specify learning goal
Identify tasks within the game
Base design on constructivist theories
Provide immediate feedback
Provide help / explanation facilities
Provide scaffolds/ support

Table 2.5 Summary of Game Play Conceptual Design Considerations

Design challenging games
Design a coherent world
Consider real places/ people/ culture
Link metaphors with learning objects
Design to allow coding practice
Provide scenarios adapted from real life

Table 2.6 Summary of Empirical Evaluation Studies

Study	**Subjects**	**Number of participants**
Osborn et al. [195]	University and college students	76
Machiori et al. [150]	University students and educators	30
Khenisi et al. [117]	Tertiary school students	167
Anderson et al. [13]	Undergraduate students	43
Gaeta et al. [76]	University students	20
Eaachak et al. [72]	Children	5
Pez-Colado et al. [210]	Teachers, artists and programmers	5

CHAPTER 3

Understanding User Needs and Requirements

3.1 Introduction

This Chapter presents three studies conducted to understand the needs and requirements of higher education CS1 educators. First, topics that novices find most difficult in CS1 were identified in a preliminary needs assessment study. This formed the basis of the problem. In the second study, potential users were given an opportunity to express their requirements and expectations. Lastly, study three investigated three visual approaches of teaching recursion with games. From these studies, we elicited requirements for designing the prototype game generator tool architecture. Results provided an opportunity for the ideation and conceptualisation of game design solutions to the identified problem. Contributions of this Chapter are:

- It presents studies on teaching CS in the context of developing countries that could be extended to other parts of the world.
- It highlights factors affecting teaching the recursion topic via games and suggests practical ways to address them as well as suitable teaching analogies.
- Theoretically, it proposes design principles that could guide the creation of programming serious games to teach diverse learners in CSE.

The rest of the Chapter is structured as follows. Section 3.2 presents details of problem identification and a needs assessment study. Section 3.3 outlines a conceptual study conducted with 17 CS1 educators to elicit user requirements. Lastly, Section 3.4 summarises a study on three visual ways of teaching the recursion topic with games.

3.2 Study 1: Preliminary needs assessment

This study identified the most difficult CS1 topics and problems novices face when learning programming. The immediate problems/ needs of introductory programming lecturers from Kenya and South Africa when teaching novices CS1 were singled out. In middle-income and low-income countries, it is not necessarily the case that students

will have the same educational background and so may learn differently. Supporting this argument, Pillay and Jugoo [201] indicated that students whose first language was the same as the language of instruction (English) performed better in programming than those whose mother tongue was not English in a South African study. Dlodlo [191] through a study of a rural village in South Africa found that language barrier, shortage of equipment, lack of Internet access, shortage of Information and Communication Technology (ICT) teachers and experts, inadequate access to hardware and software, and illiteracy were among factors that hindered ICT education for girls and women. Realising the challenges students who have poor background in computational and algorithmic thinking face in learning Computer Science, Barr and Stephenson [28] advocated for the inclusion of these skills in the K-12 syllabus.

People learn differently in different circumstances. However, we cannot predict how students learn when the circumstances are different, therefore we cannot assume that the topics they are going to find difficult to learn are going to be the same in programming. It was therefore hypothesized that, because of their circumstances, students from middle-income and low-income countries would find introductory programming courses far more difficult compared to others from other parts of the world. Meanwhile, there is growing interest in the CS community to understand the issues and difficult topics for novices. Many works have indicated that recursion and parameter passing are difficult topics for most students [124, 10, 280]. There is a consensus in the works of Ambrõsio et al. and Malik and Coldwell-Nelson [10] that recursion is the most difficult and misunderstood concept. Another topic considered difficult is arrays [10, 280]. Though identifying student's misconceptions and difficulties is regarded as a key computer science teacher's competency, relevant research on this topic is not fully developed in the CSE field compared to mathematics and science education [211]. Therefore, the preliminary needs assessment study sought to answer two questions:

1. *What do lecturers from middle-income and low-income countries perceive as the topmost issue that makes learning introductory programming difficult among novices*?

2. *What syllabus content is most difficult for novices and is it the same as for other students from other parts of the world*?

3.2.1 Methods

Ethical clearance was sought from the Faculty of Science Research Ethics Committee of the University of Cape Town. The purpose of the study was explained to potential lecturers teaching introductory programming courses during the annual Southern African Computer Lecturers' Association (SACLA) 2018 conference. Randomly sampled lecturers were requested to share their contacts and years of experience in teaching the

course. The other participants from Kenya were recruited through snowball sampling. In phase one, in-depth interviews were conducted with five experienced lecturers who had taught the course for more than 10 years. One of the lecturers had taught introductory programming courses for 14 years, three for 13 years while another for 18 years. In phase two, additional data was collected through an independent online survey with 30 lecturers. Though the sample size could be considered small, the study only targeted lecturers who had taught introductory programming courses in the recent past.

In a previous study, a questionnaire was used to identify priority topics for developing a program visualization environment for students [44]. The online questionnaire used was informed by this prior work. In phase one of the study, one of the items asked the five experienced lecturers to mention the problematic issues as well as topics they felt were most critical to teach in the course. These were used to identify the candidate topics and issues to include in the online questionnaire. A summary of some of the topics follows:

(i) Programming basics.

(ii) If statements.

(iii) Loops.

(iv) Arrays.

(v) Functions.

(vi) Recursion.

(vii) File and exception handling.

On the issues making the course challenging and difficult to learn by novices, the following were reported:

(i) Misconceptions about computing.

(ii) Lack of critical thinking skills.

(iii) Lack of problem solving skills.

(iv) In ability to design algorithms.

(v) Lack of confidence.

(vi) Lack of debugging skills.

The above topics and programming issues, among others, were considered and refined then used to design expert interview guide questions (See Appendix A) and an online questionnaire (See Appendix B). Topics that lecturers deemed most difficult for novices in introductory programming were investigated. Additionally, issues that make learning programming difficult were explored. For data analysis, Thematic Content Analysis (TCA) was done on the collected qualitative data. Open coding was used to organise the interview data to come up with initial categories. Axial coding was then applied to establish the relationships between the initial open codes. Further analysis of the axial codes was then conducted to single out final selective codes. Additionally, descriptive statistics was used.

3.2.2 Result 1: Demographic Information

This Section presents the demographics of the 30 participants in the second phase of the study. When grouped by region 16 (53%) came from Kenya. Another 14 (47%) were from South Africa. Regarding work experience in-terms of the the number of years they had taught CS1 courses, five (17%) reported less than three years, seven (23%) 3-5 years, 11(37%) 5-10 years, six (20%) 10-15 years. Finally, one respondent said that they had taught introductory programming courses for over 15 years. It is worth noting that most of the lecturers had between five and ten years of teaching experience. Regarding the programming language, participants were allowed to choose more than one language which they had used in the recent past to teach the course. C++ was selected by 16(53%). This was followed by Java 15(50%) then C 13(43%). Other languages (Visual Basic, C#, SQL and PHP) was selected by another eight participants (27%). Lastly, Python language was chosen by six (20%) lecturers - all from South Africa. Although the work by Koulouri et al. note that teaching introductory programming with a language with simple syntax such as Python improves the ability of students to learn programming concepts [123], the results suggest that many CS1 teachers from Kenya and South Africa still use languages which are perceived to present difficulty with regard to syntax and semantics to novices such as C++, C and Java. A summary of the demographic information of participants of the second phase of the study is presented in Table 3.1.

3.2.3 Result 2: Programming issues

Regarding issues that make programming difficult among novices, content analysis of qualitative interview data revealed that problem solving was the topmost issue. Direct phrases from respondents that pointed to this theme represented 48%. This was followed by algorithm design (20%), student background (14%), teaching approach (10%), and debugging skills (8%). Among the respondents surveyed, 80% indicated

Table 3.1 Demographic information

Variable	Category	Freq	Percentage %
Country	Kenya	16	53
	South Africa	14	47
Experience	Less than 3 years	5	17
	3 - 5 years	7	23
	5 -10 years	11	37
	10 - 15 years	6	20
	Over 15 years	1	3
Language	C++	16	53
	C	13	43
	Java	15	50
	Python	6	20
	Others(Visual Basic)	8	27

that problem solving was either difficult or very difficult suggesting that it was a problematic area. This was followed by 'programming logic', 'debugging'and 'translating logic to code', with 70% for the total of very difficult and difficult in each case. A Summary of descriptive statistics data is presented in Table 3.2, where 1 means very difficult, 2 difficult, 3 neutral, 4 easy and 5 very easy.

Table 3.2 Issues making learning programming difficult for novices

	1	2	3	4	5
Designing Algorithms	22%	33%	30%	13%	0%
Using IDE	0%	17%	37%	37%	10%
Debugging	20%	50%	23%	7%	0%
Programming logic	10%	60%	27%	3%	0%
Syntax and semantics	13%	40%	30%	17%	0%
Control structures	7%	43%	40%	10%	0%
Problem solving	33%	57%	10%	0%	0%
Translating Algorithms to code	23%	47%	23%	7%	0%
Tracing through code	20%	47%	27%	7%	0%

3.2.4 Result 3: Difficult CS1 Topics

Following Thematic Content Analysis (TCA) of key phrases and words from survey responses, four subcategories emerged: (i) recursion (ii) loops (iii) arrays and (iv) functions. Phrases that depicted recursion as a difficult concept to learn appeared 22 times. Words and phrases indicating loops appeared 13 times, followed by arrays nine times and, lastly, functions seven times. As such, content analysis of qualitative interview

data on the perceived difficult topics placed recursion on top with 43%, followed by loops (25%), arrays (18%) and functions (14%). Meanwhile, results from the survey on the same question showed that recursion and Abstract Data Types were considered the most difficult topics for novices. Table 3.3 presents a summary of the results, where 1 means very difficult, 2 difficult, 3 neutral, 4 easy and 5 very easy.

Table 3.3 A summary of results on difficult CS1 topics

	1	2	3	4	5
Variables and algorithms	0%	7%	23%	50%	20%
Data Types	0%	10%	23%	53%	13%
Expressions and semantics	3%	17%	23%	43%	10%
Input and output handling	3%	13%	33%	43%	7%
Sequential structures	0%	27%	47%	23%	3%
Looping structures	13%	43%	30%	13%	0%
Arrays	23%	67%	10%	0%	0%
Function definitions	13%	50%	30%	7%	0%
Parameter passing	30%	50%	17%	3%	0%
Recursion	57%	33%	10%	0%	0%
Abstract Data Types	57%	33%	10%	0%	0%
Files	27%	53%	13%	7%	0%
Testing	33%	37%	30%	0%	0%
Recursion	20%	60%	10%	10%	0%

3.2.5 Discussion

Results from this study suggest that recursion, arrays and abstract data types are considered as difficult topics for most novices. Recursion is believed to be the most difficult topic, as also reported in some previous works [146, 124, 280]. On the other hand, problem solving is identified as the topmost issue that makes learning programming difficult among novices. Given these findings, we argue that despite the different learning situations, topics traditionally considered difficult for novices universally remain the same in middle-income and low-income countries as in other parts of the world. Consequently, as a case study in this thesis, the recursion topic was used to test the idea of a serious game generator tool to support CS1 instructors.

3.3 Study 2: Requirements Analysis

Following the successful identification of the problematic CS1 topic in the preliminary needs assessment study, a second user study was conducted to analyse requirements from higher education CS1 teachers (the potential users of the proposed approach of a game generator tool). The main objective was to investigate key conceptual design

aspects that could be considered when developing a prototype of such a tool. The study also looked at the aspect of diversity in the design of games as an intervention in CSE.

The debate on diversity in CSE is not new. The early work by Camp [46] brought to the attention of the community the declining number of female graduates in CS, pointing out the urgent need for gender inclusivity. Further, the community was called upon to come up with innovative ways to attract and retain students, particularly female, in CS to address the shrinking pipeline. Reiterating low enrollment rates and the lack of diversity in Computer Science education, Bruckman et al. called for changes across the CS pipeline [43]. In South Africa, some works have recognized the diverse nature of most classrooms as demonstrated by efforts aimed at preparing lecturers to cope with and handle students from culturally and economically divergent backgrounds [25, 222]. In the requirements elicitation study, diversity refers to differences among students due to race/ gender as well as diversity in: (i) motivations, learning needs, educational level; and (ii) academic/ skills as well as cultural and economic backgrounds [176, 257].

Lecturers of CS1 encounter diverse multitudes of students [176, 257], bringing with it many challenges [176]. Guzdial and Soloway reiterate that most of these students - the Nintendo or MTV generation - are not easy to motivate using invisible and abstract traditional teaching approaches[89]. As such, using old teaching techniques may not be beneficial when engaging with them in the classroom. Probably, new approaches such as digital GBL could be worthwhile. To this extent, some works have suggested the use of games to motivate this generation of students who apparently like engaging with multimedia and animations [89] as well as games [148]. For instance, in South Africa, given the Apartheid system and the diversity of this generation of students entering universities, a pedagogical design for teaching novice programmers CS1 within the local context was proposed [51]. One of the design principles involved students using construction tools such as puzzles and games while interacting in small groups and speaking in their local languages - mainly Zulu. The results seemed promising [51].

Despite such early efforts, there still appears to be no consideration for diversity in serious games design. This was echoed in a report by a special working group on game development for CSE that noted the significant design challenges as well as opportunities presented by different learners given varied gaming backgrounds [108]. If the motivational need of the 21st century learner is to be met through the GBL approach then design needs to cater to diverse students, especially in CS1 courses. Given these revelations, coupled with the well known difficulty of the recursion topic among students [17, 124, 160, 172, 234], the purpose of this study was to identify the needs of CS1 educators and particular groups of learners with an attempt to understand how programming games that appeal to diverse learners could be designed. The study had three specific objectives:

1. *To find out the teaching practices used by programming lecturers in Kenya (low-income country) and South Africa (middle-income country) when teaching undergraduate novice students the recursion topic.*

2. *To investigate the examples/ analogies suitable for designing games to teach the recursion topic to diverse novice students.*

3. *To explore conceptual design considerations deemed useful when developing a game authoring tool to create games that can teach diverse novice students the recursion topic.*

By answering these questions, the novelty and hence contributions of this particular study are as follows:

1. *It is a unique study on teaching CS in developing countries that could be extended to others.*

2. *It highlights factors affecting teaching the recursion topic via games and suggests practical ways to address them as well as suitable teaching analogies.*

3. *It proposes design principles to guide the creation of programming games to teach diverse learners.*

3.3.1 Methods

The three objectives were addressed by interviewing 17 CS1 lecturers from higher learning institutions based in Kenya and South Africa. Four participants were female and 13 were male. Of the females, three came from South Africa and one from Kenya. Among the males, five were from Kenya and eight from South Africa. Twelve respondents (70%) were very experienced. Overall, the most experienced had 30 years and the least three years of experience. One of the participants, a professor and an expert in game development, was recruited on purpose to give insights from the game design and development perspective. Six respondents had used games in teaching. Four of the six (66%) had personally designed the games used. Consequently, the data gathered was considered relevant from both pedagogy and game design perspectives [260]. All the participants from South Africa were recruited from the Southern African Computer Lecturers'Association CS1 lecturers' list created in 2018. On the other hand, it was convenient to recruit respondents from Kenya because of the already established contacts.

Ethical clearance was first sought from the Faculty of Science Research Ethics Committee of the University of Cape Town and relevant institutions. Invitation emails were then sent out explaining the purpose of the study to the potential participants. Thereafter, interviews were scheduled with the 17 willing participants. This was followed by

sending written informed consent forms for signing before the interviews. Each interview lasted an average of 45 minutes. The interviews were structured to have focused topics aimed at eliciting particular themes relevant to the focus of the research. The interviews contained questions about the teaching experiences, game adoption barriers, teaching practices (e.g. teaching time and class examples) as well as the recommendations on suitable teaching analogies for designing games to teach the recursion topic to diverse students. Additionally, there was a question regarding what the lecturers thought could guide the design and development of a game authoring tool capable of creating different games for diverse students (See the interview guide for teachers in Appendix B).

As the interviews progressed, recurrence of similar data started to emerge. By the final participant, data collection reached a point where no new insights were generated i.e. data saturation was reached [248]. At this point, no new participants were recruited. Skype interview sessions were recorded using FlashBack Express video screen recorder. For the respondents interviewed face to face, recording was done using a mobile phone audio recorder. A game expert validated the findings. Regarding analysis and presentation, findings were manually transcribed and summarised into textual data using Microsoft Word. This was followed by TCA [15, 245, 273]. TCA is a descriptive presentation of qualitative data such as interview transcripts gathered from respondents or other identified texts on the topic of study [15]. Common concepts/ ideas in the textual data were identified/ highlighted and grouped into common categories/ themes. Codes (concepts) were either included or excluded guided by the interview questions. Coding was iteratively done until no new themes emerged. Overall, three iterations were done. The first author and the supervisor who conducted the analysis met regularly to discuss and agree on the coding/ analysis of data. Where appropriate, tables were used to guide discussions.

3.3.2 Finding 1: Game Adoption Barriers

After asking respondents about their demographic information, an inquiry was done to find out potential reasons why games have not been widely adopted in teaching. Eighty two percent of the respondents felt that lack of time in the CS1 curriculum was the main reason why the adoption rate of the GBL approach was low in mainstream teaching. Other reasons given were: lecturer background and lack of awareness (59%); difficulty in getting relevant well - designed games and tools (53%); and uncertainty with the GBL teaching approach to deliver content (18%). Some remarks follow:

> *"It is hard to infuse games into the curriculum considering time".*

" In my programming education up bringing I have never seen games used in teaching ... so the way I was introduced to programming is what I give back".

"when I was taught programming in my university I never learned with games" .

"I do not know how effective it would be compared to the traditional learning and so I fear trying to use it because I am not sure of the impact".

"It takes time and again there are no right game resources".

These findings are consistent with results from some previous works [98, 254]. In a broader sense, the findings on adoption barriers could extend the debate on the diversity topic by stressing the need to expose students to introductory programming [54], and use of alternative learning media such as games in an attempt to address the shrinking pipeline problem in STEM.

3.3.3 Finding 2: Teaching Practices

Regarding current practices used by lecturers when teaching novices the recursion topic, three issues were investigated. These were: (i) teaching time; (ii) diversity and size of CS1 classes; and (iii) teaching examples. It emerged that the CS1 curriculum in most universities in Kenya and South Africa does not allocate enough time for the recursion topic. A majority of the participants reported spending from three to four lessons only on the topic in a semester. This was considered insufficient given the topic difficulty. As an illustration, one lecturer mentioned spending from two to three lessons - each being 60 minutes. Another said from five to seven lessons, each 45 minutes. Meanwhile, two others reported three to four lessons, each 80 minutes. Lastly, three respondents reported three lessons, each 45 minutes. While acknowledging the importance of recursion in computer science, some participants alluded that the CS1 curriculum did not allocate much time for the topic. Consequently, they were forced to find extra time to reinforce the concept. Comments from such respondents included mentions like:

"... in our current curriculum we spend only two or three lectures if ... may be only two but not much".

"I think it is always sort of the pillar of all these programming courses ... you find it is the one always you repeat after doing ... students will always want you to come back and back and back and back from the first time you introduce it".

"I usually find time and come back to it after teaching binary search and sorting just to try and reinforce it".

Given the pressure of time, two models of practical game adoption were proposed: (i) short games for quick just in time class illustration; and (ii) long games for after class practice. One lecturer proposed a design approach that could give students games that allow them to practice for longer hours after class while at the same time give a lecturer the ability to track their progress. The participant made particular reference to universities that enroll large numbers of students, for instance from 700 to 1000, for CS1 with different backgrounds. This proposal was supported by another respondent who added: *"there was a time we wanted to change one of our introductory course ... only to realise that over 70% of the students were government sponsored with no computer background"*.

When it comes to diversity and size, most respondents confirmed that the practice is that typical CS1 classes attract large numbers of diverse students [9] drawn from different departments. The diversity is manifested in the different academic, economic and cultural backgrounds. One respondent from South Africa reported that her university enrolls very many students in CS1 classes. In particular, the number was reported to range between 700 and 1000. Given the large number, she reiterated: *"with large classes it is not easy to have a way of tracking student participation"*. Another respondent from Kenya reported that the course attracts 200 and above students per class every Semester it is on offer. For better student to teacher ratio, this number is divided into 40-50 students per class and assigned to different lecturers. Concurring with this practice, another respondent from South Africa who emphasised the diversity of students reported that his university attracts about 80 students in CS1 classes. This is usually divided into two groups - 40 students each. Another practice is the use of teaching assistants to manage smaller groups of diverse CS1 students during practical lab sessions.

The finding on time is relevant and in agreement with that of Battistella and Wangenheim [29] on games for teaching computing in higher education that showed that most games are designed to fit into the duration of a class or to be played rapidly (about 10 minutes). Additionally, other than the practical aspect of time, these results further supplement current literature on programming game design by introducing the practically large number and diverse CS1 students aspect.

Next, participants were asked to comment on the examples they use in class to teach recursion. Findings indicate that a majority (77%) use textbook examples. Popularly mentioned were: Factorial, Fibonacci series, Greatest Common Divisor (GCD), binary search, power of a number, palindrome, strings, towers of Hanoi, and recursive patterns. It was surprising to note that only two lecturers (12%) said that they use real-life scenarios in class. Similar to findings from a recent study in USA that investigated the teaching practices of introductory CS course instructors [97], results of this study also revealed that CS1 teachers from Kenya and South Africa use instructor-centered teaching practices. Such practices are commonly dominated by lectures instead of student-centered

activities. Consequently, the practices undermine students'conceptual understanding of course content [44].

3.3.4 Finding 3: Teaching Analogies and Design Ideas

When asked to recommend suitable teaching analogies/ examples that could be designed in games to teach novices the recursion topic, fifteen respondents (88%) agreed that simple algorithms should be used. Simple mathematical examples such as factorial, GCD, power of a number and Fibonacci series were the most popularly cited. Participants warned of the trap of adding extra cognitive load in the design by introducing complicated examples. As a classic exemplar, one lecturer was reported saying: *"let them be simple ... not very complicated/ difficult algorithms ... I tried using a normal fan like pattern in my CS1 class but from experience students found it difficult to understand recursion"*. A mixed reaction was noticed on whether to use 2D or 3D games. While a lecturer who also teaches a design course and another (a professor in game design) proposed a 2D game with simple and neat rewards arguing that it will keep it simple and reduce distraction, another lecturer with an interest in graphics recommended 3D for their visual learning appeal.

Though textbook examples were suggested by most participants, some respondents brought up a useful debate on the extent to include them. One respondent felt that it was mandatory to fully include them by arguing: *"classical recursive examples like factorial, binary search, Fibonacci must be fully there in the design knowing very well the importance of maths in computer science"*. On the contrary, another saw it differently, commenting: *"I think their use is a disadvantage to students with poor mathematical background"*. This view was supported by another respondent who also cautioned against using only mathematical examples, particularly, those that novice students had not encountered in their high school. The participant said: *"for me the expectation that every student entering university understands factorial is reasonable ... expectation that they will understand Fibonacci am not sure, I don 't think that much thought has gone into the possibility of using real-life scenarios"*.

Findings on class examples and proposed teaching analogies, potentially, could equally add to the literature on the topic of diversity in CSE. It seems that inclusivity in programming games design could be advanced by using a variety of teaching examples that would cater for different learning needs. This is in line with the recommendation by Ibe et al. [101]. It could therefore be argued that a design approach that adopts both real life and textbook examples would increase the potential of diverse CS1 educators and learners to embrace a GBL approach.

3.3.5 Finding 4: Design for diversity

In the last section, respondents were asked to mention what they thought could guide the design and development of a prototype game authoring tool capable of creating different games for diverse CS1 learners. To elicit this information, three aspects were examined: (i) unique design factors; (ii) customisable game attributes; and (iii) pedagogy.

3.3.5.1 Unique design factors

From the text analysis of the responses received about unique factors that lecturers thought could guide the design of programming games for diverse students, four key themes emerged, namely: (a) context; (b) gender; (c) religion; and (d) childhood game experience.

It was the considered opinion of some participants that it is important to factor in local context when designing serious games for novices (particularly from Africa). To illustrate this, three lecturers from Kenya and one from South Africa contributed to the debate by arguing that people from different parts of the world were sensitive to different icons, shapes, and symbols, based on their cultural backgrounds. Of the three, one (a relatively young participant in age) could not hide the frustration with the fact that most digital games played in Africa borrow a lot from the West. The respondent lamented saying: *"why not contextualize to local games such as ... 'Banu'and local characters"*. Banu, also known as 'Banta'in Kenya, is a marble game commonly played by two people during childhood. Another Lecturer from Kenya who has taught a multimedia course using games also suggested a localised game scenario where a Masai Moran (warrior) fights lions to protect a Manyatta (hut) with the warrior as the main character. Masai is a well known community in Kenya for maintaining their culture. A similar remark came from one more participant from Kenya whose opinion was: *"if you are to design a game for the youth in Kenya then probably, characters based on local games, football stars, celebrities in music industry, and surprisingly politicians will make them interested"*. Finally, a respondent from South Africa echoed the similar sentiments by saying: *"background and culture will affect the icons ... shapes ... symbols that you use ... people from different regions are sensitive to different symbols ... a certain icon in Cape Town will mean a totally different thing in Kenya"*. Consequently, it was considered key to adopt localised game scenarios and characters in design.

On the issue of gender, 18% of the respondents felt that it is important to consider gender in the design of serious games to teach novices. For instance, after the interview, one respondent further sent an email suggesting two different design scenarios for male and female students. This finding is in line with a previous study that reported that women and underrepresented groups are often turned off by competitive games [271]. The work noted the need to lay more focus on socially-constructive applications.

Adding his voice to the gender debate, another participant (a professor and an expert in game programming) suggested a gender neutral design approach.

Meanwhile, results from four (24%) participants revealed that the games learners played while growing up could also have an influence on the type of serious games that would interest them in CSE. One of them described the experience when teaching an introductory programming course in the Northern hemisphere. Classes started in October where students did an objects first introductory course from October to December. This was followed by a project in December during the last week of the semester. Students were allowed to pick a project to work on based on scenarios they were interested in. The respondent reported: *"one year, two of my students collaborated in a battle Bot game ... a game of Robots ... and they really enjoyed the fact that it was something they could relate with remembering playing it when growing up"*. Another practical demonstration on how childhood game experience could influence game design for different students in CSE was a revelation by a lecturer from Kenya. The participant recalled vividly one particular teaching experience by remarking: *"...I remember there was a time when we had some students actually using AI techniques to develop a local game ... we picked this game so that students could relate with things they have grown up with..."*. In one unique case, another respondent suggested that lecturers should be careful when using games that include gambling since they may offend some religious beliefs. As in the National Science Foundation [187] report, the findings on gender, context, religion and childhood game experiences contribute to the diversity topic by considering target learners, but from Africa. Particularly, in the context of South Africa, universities predominantly enroll CS1 students from diverse racial, cultural and computer knowledge backgrounds.

3.3.5.2 Customisable game attributes

Participants were further asked to propose game attributes or features a lecturer could change/ customise to create different instances of games with unique game play experiences using a game authoring tool. Pointing out the lack of such tools, one respondent said: *"I have not seen any educational game development software"*. Supporting this, two other participants observed that game generation would help lecturers save on time spent searching for appropriate teaching games and also present them with variety. One of them noted: *"a number of years if you keep using the same examples over and over the students don't see the value ... there is need for refreshing examples used by lecturers"*. A similar view was held by another who remarked: *"do not constrain them to a particular example"*. The two views illustrate the need for variety.

Game complexity and challenge levels was also suggested by sixteen (94%) respondents. Some comments included: *"design for challenge at different game levels"*;

"different levels of difficulty/ complexity-for different types of students"; *"change game complexity"*; and *"vary the levels of complexity such that they start by coding simple ones and also to take care of slow learners"*. Eight lecturers felt that characters and their unique parameters like clothes and colour could also provide an opportunity to create unique game experiences. While backing up the idea of game generation, one respondent who considered himself a gamer said: *"I think a nice feature would be a game that has a lot of customization or configuration options for instance I remember in the car race game you could change it to your specification ... change the color, tyres"*. On the contrary, while stressing the role of visual learning, another respondent observed that the design should steer way from using characters since such could bring in personal issues. Instead, it was advised that neutral objects could be used in their place. Other customisable game features suggested were: scene, background, theme, texture, light intensity, interaction style, genre and character. A summary of some customisable game features is presented in Table 3.4.

Table 3.4 Customisable Game Design Attributes

Game Genre	**Game scenario**	**Back ground**	**Character**	**Interaction style**	**Complexity**
Adventure	Maze type game	Color	Cloth	Drag and drop	Simple
Role Playing	Factorial of a number	Scene	Hair	Write full code	Intermediate
Simulation	Simple pattern	Texture	Body color	Complete snippet	Complex
Action	Tree traversal game	Sound	Eye color		

3.3.5.3 Pedagogical considerations

Finally, participants were asked to reflect on their teaching experiences and share pedagogical/ instructional design considerations they deemed suitable for games to teach recursion to different students. It emerged that there is need to balance game play and learning given the diversity in a CS1 class. Two lecturers were quick to caution that from their experience one of the reasons why lecturers do not use games in class and some students do not take them seriously is the fact that they do not directly see their educational value. One went ahead to advise: *"design to engage them in such a way that they see relevance ... value in it"*. This was supported by another who added: *"students may feel they are just playing if the feel of playing to learn is not there, particularly the bright ones"*.

Twelve (70%) respondents pointed out that individual differences among learners in a class ought to be considered. One remarked: *"I tried using a game in class and the challenge was to get tasks to take care of the weak students ... pitch a game at the very weak students and their capabilities ... design for different types of learners"*. Another thirteen participants (76%) made remarks suggesting that game goal and learning goal ought to be clearly aligned [209], and that a deliberate effort should be made to balance design to cater for all types of learners. In CS, the Cargo Bot serious game by Tessler et al. [255] and Program Your Robot by Kazimoglu et al. [113] are illustrations of design by aligning learning and game goals.

Two lecturers felt that programming game design should allow students to actively construct their programming knowledge through active coding. For instance, one said: *" my experience is that it is important to let students be in control of their learning by writing code"*. This position was supported by another respondent who commented:*"..design to take them through step by step and to practice by forcing them to write and compile code..."*. An example of a game in CSE that enables students to learn programming by practicing coding is Elemental the Recurrence [48]. These findings suggest consideration for the constructivism learning theory [12, 90] in design, particularly for diverse learners in CS1 [257].

Careful scaffolding and student support, especially for weak students, was yet another admirable instructional design feature according to one participant who said: *" design for user guided inquiry learning to allow them to try different things"*. Scaffolding is demonstrated in the Saving Cera serious game by Barnes et al. [26] and Elemental the Recurrence by Chaffin et al. [48] in CSE. Additionally, embedding motivation, reward and challenge in the design was seen as important by three respondents. The first remarked: *"I have taught second year course with games ... students were very interested ... make it fun and amazing ... amusing ...engaging...challenging...motivating"*. Singling out the use of leader boards and points, the second who had used games in teaching a third year computer science course additionally stressed the need for competition and rewards in design. Agreeing with this position, the third (a professor and game expert) added his voice by saying:*" reward system is very important from game design point of view ... furthermore the core aim is pedagogy ... to design educational games, particularly to teach the recursion topic"*.

3.3.6 Emerging conceptual design principles

Evidence presented in the requirements study suggested that the game adoption rate in mainstream teaching still remains low despite the potential motivational and retention impact on learners. Although most lecturers use textbook examples in class, it appears this could also potentially disadvantage some students and further propagate the lack

of diversity among CS students. An educational game design approach that strictly adopts only such examples might in itself cause a learning barrier to some learners. Consequently, when designing CS1 educational games for diverse learners, it would be advisable to use textbook examples learners are familiar with, simple algorithms and real life analogies. For example, the DFS algorithm used in Chaffin et al. [48] would be considered complex for CS1. In addition, the ability to have unique characters is equally important. However, caution should be taken and where necessary, neutrality could be considered in design. Some scholars have reiterated the lack of design guidelines/ principles to create serious games [52]. The requirements study attempted to address this gap by proposing eight design principles that could guide requirements engineering and creation of diverse games to teach different students programming. These include:

(i) Use diverse simple mathematical examples learners are already familiar with to reduce cognitive load, while factoring in practically available time in the curriculum.

(ii) Use a variety of fun and enjoyable real life analogies that students can easily relate with.

(iii) Consider particular gender needs.

(iv) Carefully factor in student's background in terms of computer knowledge, culture and local contexts.

(v) Carefully choose game genres bearing in mind religion.

(vi) Consider students'childhood game experiences.

(vii) Pitch the design for different learning needs of students in a class, particularly the very weak.

(viii) Design for game customization to produce a variety of game experiences.

The proposed design principles are supported by findings of other related works. For instance, while calling for a more inclusive design approach, participants in the study by Rankin [215] criticised the game design for lack of (i) diverse game characters that consider gender (principle 3); (ii) cultural aspects (principle 4); and (iii) ability to customise the game play experience (principle 8). Similarly, findings from the work by Khalifa et al. [116] confirm that culture (principle 4), customization and game attribute configuration (principle 8) are crucial design considerations when creating diverse games targeting different audiences. Moreover, two key findings from another study on the views of 41 game designers and practitioners agree with principles 2 and 7 [106]. In this study, fun is identified as a key design feature for serious games for

diversity in line with principle 2. The second key finding suggests that design should gradually release information and allow practice and mastery that can be built upon to give a sense of progression (principle 7). This supports individual progression of students in a CS1 class given different learning abilities [176]. On the other hand, the work by Katz [112] presents seven broad Universal Design (UD) principles for an inclusive education that considers diversity. The first principle - equitable use - maps to design principles 1, 2 and 3 in this paper. The second - flexibility in use - can be mapped to the proposed principles 4, 5, 6 and 7. Lastly, simple and intuitive use - the third principle - aligns to the proposed principle 1. Figure 3.1 presents a summary of the conceptual design principles.

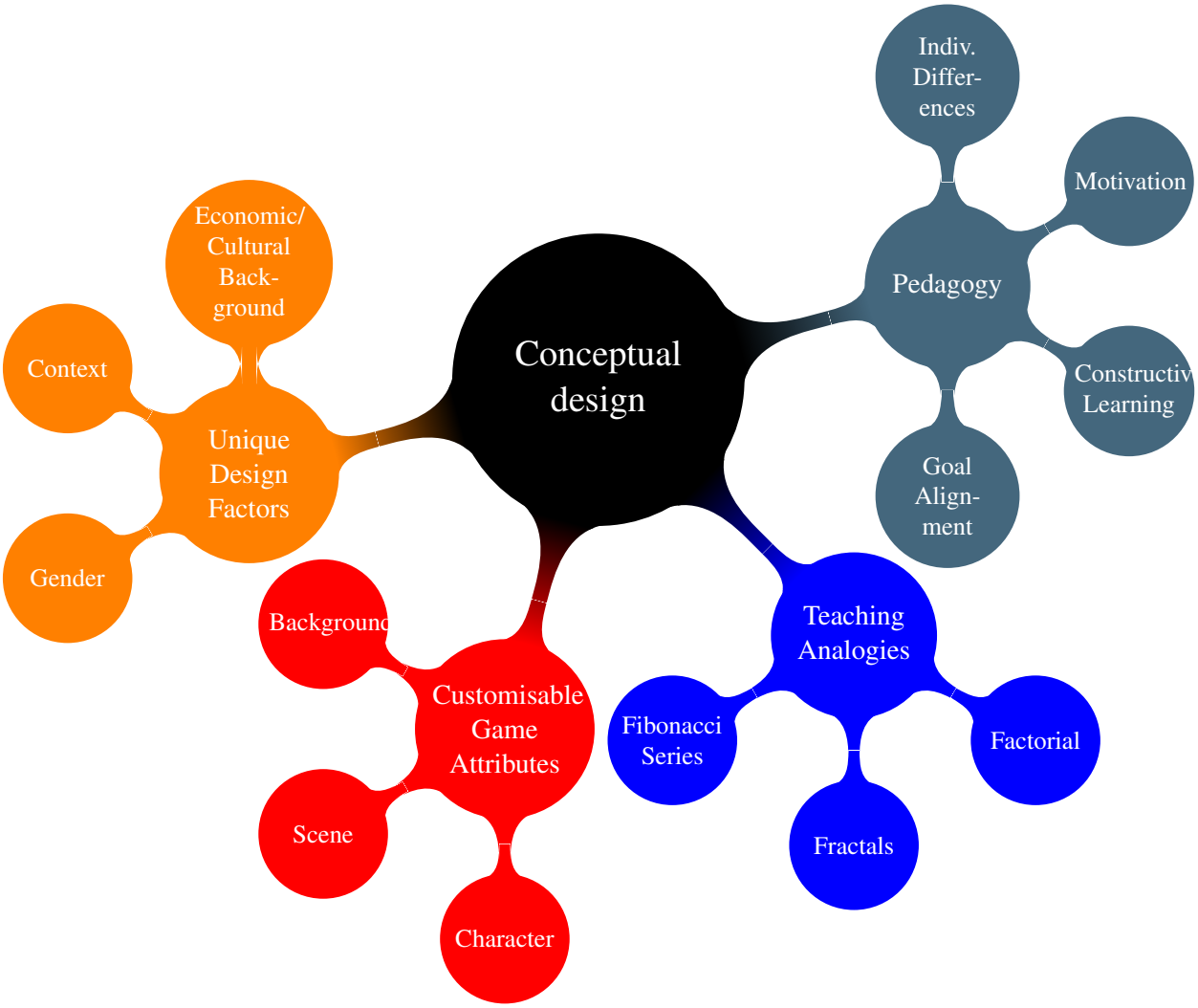


Fig. 3.1 Summary of conceptual design principles

3.4 Study 3: Three visual ways to teach recursion

In an effort to further understand additional user requirements and the prototype design aspects from the lens of students, the author supervised a CS project conducted by three honours students [1]. All three were from the department of Computer Science at UCT. The project was titled: 'Three Visual Approaches to Aid the Teaching of Recursion'. This study helped to investigate some very specific issues on usefulness of different visual elements in recursion games. The aim was to investigate three visual approaches of teaching the recursion topic with games. The three approaches were: (i) visual coding; (ii) visual simulation; and (iii) stack simulation.

3.4.1 Methods

Ethical approval was first sought from the Faculty of Science Research Ethics Committee of the University of Cape Town. Second, interviews were conducted with ten CS1 students to understand the challenges/ problems students often encounter when learning the topic of recursion in higher education. Additionally, we assessed the user requirements. Third, TCA was conducted on the interview data that led to the identification of the themes. Fourth, based on the findings from the user interviews, we designed and developed three visual game-based platforms for teaching the recursion concept to novices. Finally, the three approaches were evaluated with 20 CS1 students from the department of Computer Science in a UX study.

3.4.2 Student Interviews

Interview sessions with ten students painted a picture of the challenges CS1 students face when learning the topic of recursion. A summary is given in Table 3.5. Additionally, the respondents provided some suggestions that could be considered when designing games to teach recursion through visualisation as presented in Table 3.6.

3.4.3 Visual Platforms Design

Figure 3.3 shows the visual coding User Interface (UI). Some of the key components include the recursive question/ problem to be solved, the instructions for students to click and drag, the answer drop zones and a button to trigger visualisation. Figure 3.2 demonstrates how the selected instructions/ commands are dropped in the drop zones. This approach of visualising recursion was inspired by early programming games such as Lightbot [86] and Cargo bot [48]. On the other hand, Figures 3.4 and 3.5 illustrate the stack simulation. Some key features are the: input field, output field, function call,

[1] https://projects.cs.uct.ac.za/honsproj/2019/

Table 3.5 Analysis of student opinions about their challenges when learning the recursion topic

Theme	**Aspect**	**Sample quotations**
Time	Topic hurriedly taught	"*Too fast taught - not progressive...need more time*". "*Too little time and pressurized concept ... needs to be gradually taught*". "*Time as in solving question - need more time on assignments....need more time for concept to seep in*".
Teaching	Lecturers do not simplify and clearly explain the concept	"*Lecturers fail to explain the concept in an understandable way for the learner*". "*Not taught in an easy to understand way....it is not intuitive like loops*". "*Lectures did not explain it in a simple way....need to be giving a whole lot of exercises like converting a for loop to recursion*".
Base-case	Students fail to recognise base case	"*Difficult to visualize what's happening..I have to think of a lot of different parts like what the base case is and how the problem will get smaller with each iteration*". "*I do not know where recursion ends.....base case....do not know how it ends... concept is a blur..it is daunting*".

returns, step, and compile. The input field provides a python text editor that allows students to complete code snippets. The call stack shows the recursive calls as code execution moves towards the base case. The return shows the statements returned after each call while the step feature allows students to set breakpoints in code execution so as to visualise stack calls one by one. Motivated by study 2, the power analogy was one of the use cases adopted (See Figure 3.6). Another analogy was the factorial (See Figure 3.7). More detail on game design is in Chapter 4.

Table 3.6 A summary of some student design suggestions

Approach	Design suggestions	Sample quotations
Stack simulation	Visualising call stack	"*...We were taught to think about it like a stack of plates so maybe something like that..I would like to see the parameters passed to each call stack at the very least*". "*A stack of something like blocks or similar...smooth animations and an option to view the stack progress slowly*". "*I would like to see the status of the variables passed to the function each time*".
Visual simulation	Simple visual analogies	"*Snowball/snowman increasing and decreasing in size*". "*Some visualization about bunnies going in and out*". "*Visualization - square over other square..animal eating smaller animals...Animals come out of animal mouth*"

Fig. 3.2 Visual coding User Interface

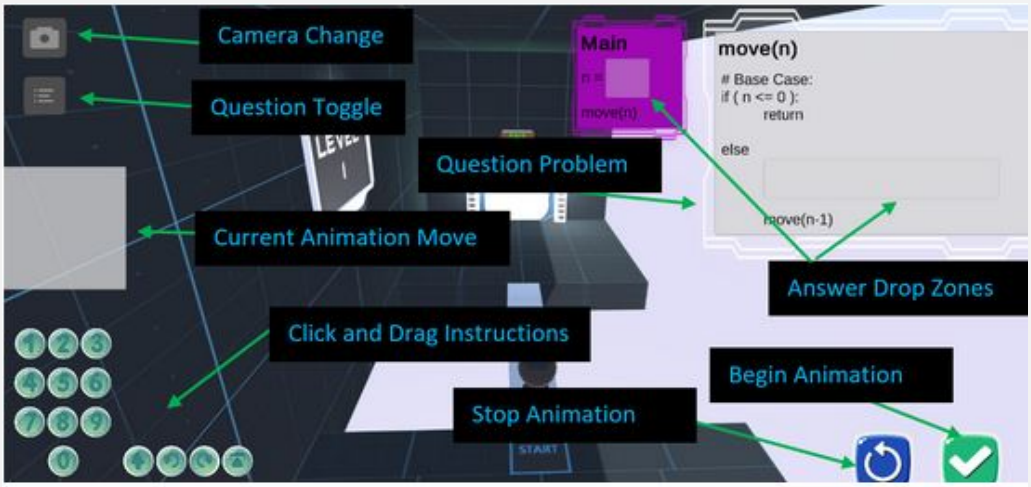


Fig. 3.3 Dropping commands in the drop zones

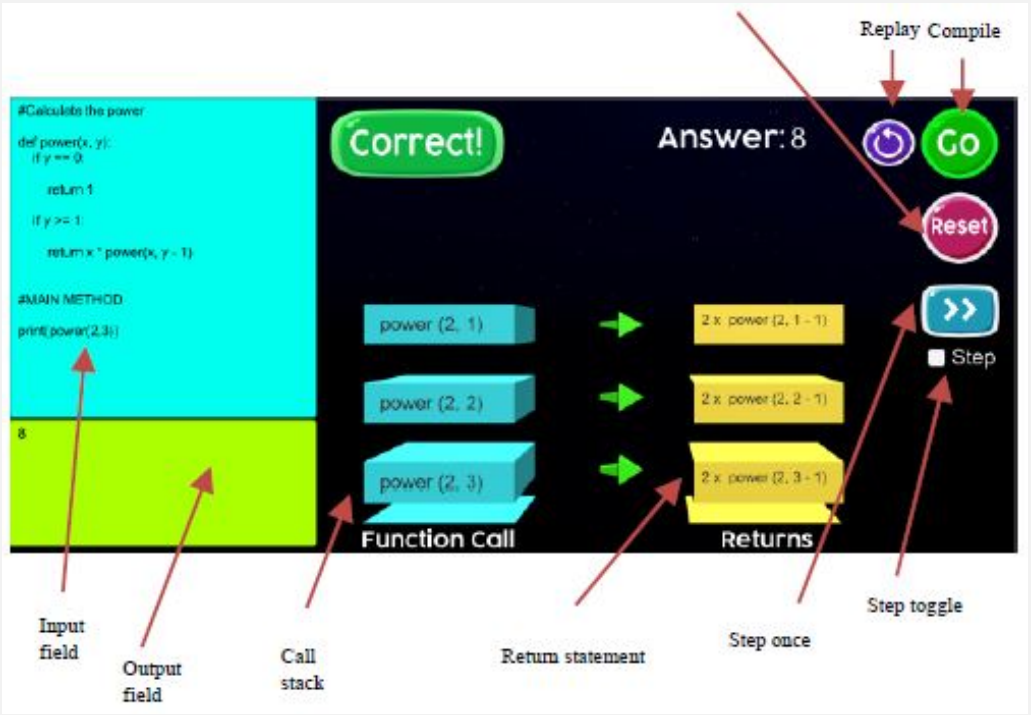


Fig. 3.4 Main stack simulation User Interface

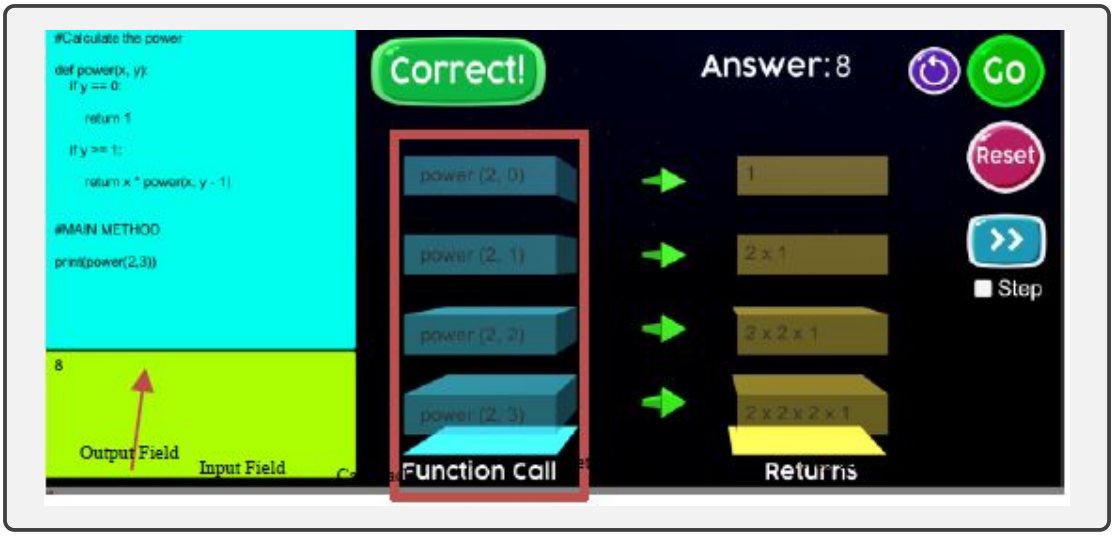


Fig. 3.5 Full built stack for recursive power

```
# Calculate the power
def power(x,y):
 if(y==0):
  return 1
 else:
    return x * power(x, y-1)

#MAIN METHOD
print (power(2,3))
```

Fig. 3.6 Power of a number use case

```
# Calculate the factorial
def factorial(n):
 if(n<1):
  return 1
 else:
    ans = n * factorial(n-1)
    return ans
#MAIN METHOD
print (factorial(3))
```

Fig. 3.7 Factorial of a number use case

3.4.4 Evaluation

Results from the Game Experience Questionnaire (GEQ) indicated that the students felt competent (mean score 3.64) when playing the designed games. Additionally, they found the games impressive, aesthetically pleasing, fun and enjoyable (mean score 4.52). Regarding the usability and UX, 95% of the students expressed that they found the visual coding User Interface intuitive and easy to use. Ninety percent of the participants reported that the text editor interface of the stack visualisation platform was easy to use. Another 80% observed that visualising stack calls using factorial was a good approach for illustrating the concept of recursion to novices. Concerning the visual simulation platform, 80% of the students found the visualizations aesthetically pleasing, interesting and appealing.

3.5 Summary

This Chapter provided an overview of three studies conducted to understand the needs of CS1 teachers when teaching the recursion topic with games and of students when learning. The needs assessment study (study 1) identified recursion as the most difficult topic among others like arrays and abstract data types. It was also revealed that CS1 topics deemed difficult among students in other parts of the world universally remain the same in poor and middle-income countries. The requirements analysis study (study 2) suggested eight conceptual design principles to guide the design of a game authoring tool capable of creating games to teach diverse novices in a CS1 class. Finally, the honours project (study 3) provided insights on how to design visual programming games to teach programming from the lens of students. Overall, these findings could be relevant for the design of future game-based learning tools in CSE. For instance, given the results, system requirements were developed as summarised in Table 3.7.

Table 3.7 A summary of system requirements

Requirements	**Functionality**
Pedagogy	The tool shall allow users to author serious games by specifying clear learning goals. Lecturers shall be able to add more recursive teaching examples. The tool shall use simple textbook recursive examples like the factorial that align with lecturers'current teaching practices. Fun learning and student motivation shall be supported in the generated games by incorporating elements such as health points, trophies, leader boards etc. Careful scaffolding shall be designed by providing help and feedback.
Game generation	The user shall be able to reuse existing game assets The tool shall generate games scenarios that consider diversity of students: computer knowledge, culture, gender needs and local contexts. The tool shall use procedurally generated scenes for more randomness. The instructor shall be able to download the generated game for offline distribution to students.
Game play	The players (students) shall have the freedom to choose any level. The generated games shall support an Integrated Development Environment (IDE) where students can actively practice coding. The generated games shall use simple rules: move left, right, up, down. After executing code students shall be able to visualise the recursive execution.
Customisation	The tool shall allow lecturers to customise game background, scene, characters etc. according to needs of different learners.
Usability	The tool shall be able to allow lecturers to create programming games easily, efficiently, effectively and with minimal cost.
Development and deployment	The tool shall be web based and capable of being deployed on a cloud server.

CHAPTER 4

Prototype Design and Development

This Chapter presents the design and development approach adopted for the game authoring tool prototype the Recursive Game Generator (RGG). RGG was created as an experimental platform. The design and development was inspired by the principles presented in Chapter 3. First, the design goal is presented in Section 4.1. This is followed by design theories in Section 4.2. Design method is summarised in Section 4.3 while prototype development is in Section 4.4. Section 4.5 presents the game generation process. Lastly, Section 4.6 is on the generated games and the features designed.

4.1 Design Goal

The design goal of this project was to develop a usable game generator tool prototype to support both novice and experienced programming instructors (realistic user audience) in high schools and higher education [181]. The target users are deemed to have little or no game programming skills to easily create serious games to teach novices programming in CS1. The design idea is to give CS educators an authoring tool with different user profiles. The prototype should allow users to easily customise given game examples to create different instances of programming games. Additionally, experienced users should be able to create and customise their own games from scratch. The proposed approach addresses three problems: (i) the difficulty encountered by teachers when creating serious games; (ii) the time constraint; and (iii) financial constraint involved.

4.2 Design theories

While foundational learning and cognitive sciences theories have been considered in the design of authoring tools, not much work has been done on theoretical foundations of authoring tools'usability [181]. Murray [181] proposed four theoretical foundations that could inform future authoring tools design, namely: (i) complexity in software design; (ii) activity theory; (iii) epistemic forms and games; and (iii) adult cognitive development theory (hierarchical complexity theory). He raised the issue of *"how*

one matches the complexity of the authoring task to the complexity of a tool and the complexity-capacity of the target user" [181, p. 3]. Complexity-capacity of a user or cognitive complexity is a "person's capacity to perform complex mental or behavioral tasks" [181, p. 18]. He stressed that the design goal should not be to develop powerful authoring tools but those that meet the needs of realistic user audiences.

4.2.1 Complexity theory

Complexity theory hides the complexities of the underlying technologies from the users (in this project: teachers). The theory supports different levels of complexity. Both novice and experienced users of a tool can easily design almost any instance of an artifact by merely selecting from given examples and customising various parameters [180]. At the same time, experienced users can access advanced system features to create complex artifacts. On the complexity issue of authoring tools, Murray [181, p. 2] poses two questions: (i) "who are going to use these tools"?; and (ii) "how do we make sure that the tools meet end user needs"? In this thesis, a balance must be created for the purpose of the target user audience. This is because, if the design only supports advanced users who may wish to add advanced pedagogical content then the tool's usability would be compromised, particularly with regard to novice users [174]. On the other hand, if design purely targets novice users who are only interested in serious games with simple pedagogical content then complexity may suffer at the expense of usability [180]. Thus, if the design is to support both types of users, a trade off is required [181, 180, 174]. Murray [180] proposes the following trade off list for future authoring tools'design:

(i) Power/ Flexibility

- Breadth (Scope)
- Depth

(ii) Usability

- Learnability
- Productivity

(iii) Fidelity

(iv) Cost

The right balance must be established between usability (simplicity and efficiency of use) and power (flexibility, breadth and depth) for the intended realistic users and final generated game instances.

4.2.2 Differentiating interfaces theory

A study conducted by Karoui [111] noted that teachers found Mobile Learning Games (MLGs) authoring tools either too poor to create games that could fit their teaching needs or too complex to use. To solve this problem, the authors proposed an authoring tool design model that supports several conceptual levels. The rationale is to design different interfaces targeting different user profiles (novice and experienced tool users). A nested design approach comprising novice, intermediate, and advanced modes is suggested [111]. With the novice mode, novice programming teachers are able to easily author simple learning games to experiment with students in class. To gain further experience with the tool, the design gradually accords such users more features at the intermediate mode. Lastly, the advanced mode allows more experienced teachers to explore advanced features when creating more complex game instances [111].

4.2.3 Design theories choice

Hidden complexity theory was chosen to guide the design of the authoring tool prototype because it supports design of simple tools for novices but also provides the flexibility to allow experienced users to explore complex system features [181, 242]. Likewise, differentiating interfaces theory was preferred because it ensures the design of different interfaces/ user profiles for different users (novices and advanced users) [111]. Consequently, the two theories were deemed relevant since the proposed tool is envisaged to support both experienced and inexperienced teachers. The inexperienced users may include trainee teachers as in Section 5.3 and those who have just commenced their teaching carrier. Meanwhile, experienced users may comprise teachers like the ones who participated in experiment 1 (Section 5.1).

4.3 Design Methods

Some previous works have explored methods to guide development of serious games [102, 226]. For instance, Ibrahim and Jaafar [102] proposed a model comprising game design, pedagogy and learning content modelling. On the other hand, Saavedra [226] suggested a software process to game design comprising five steps: (i) requirements gathering - setting learning goals; (ii) design - creating digital resources; (iii) development - creating the game; (iv) testing; and (v) postmortem. While the process by Ibrahim and Jaafar[102] is a good starting point, it seems best suited for large scale commercial serious games. This is because it ignores practical issues such as available development time and the challenge of bringing together all experts (instructors, programmers, game designers). The design process of the prototype game generator tool followed the UCD method [21, 216]. UCD was chosen since it: (i) gives useful

user feedback on the proposed design ideas; (ii) increases acceptance of the system being developed by the users; and (iii) improves usability of the system under development. Additionally, *"user centered game design recognises the benefits of play early in the design phase, allowing developers to implement these advantages into the core behaviours of the game system "* [216, p. 2]. By employing user centered methods to gather design suggestions for learning technologies, previous works have suggested that users are best involved as informants [267]. Conceptual design process in this context involves a rapid series of iterative tests and revision cycles, coupled with participation of CS1 teachers as informants.

We adapted some aspects of game design methodology from the early works by the Game Development for Computer Science Education working group of Innovation and Technology in Computer Science Education 2016 (ITiCSE 2016) [108] and Saavedra [226]. The working group highlighted some aspects of game development methods that they considered important for new researchers who may want to create education games for CS. These include:(i) Agile processes - breaking a project into tasks or sprints and monitoring at the end of each sprint what is going on well and what is not; (ii) Moodboards and storyboards - used to conceptualise the feel and look of the game to be developed; (iii) Paper prototyping - presenting a game idea through storyboards or mock-ups (Figure 4.1 and Figure 4.2 show some of the paper prototypes); and (iv) Quality assurance and user testing - done to get user feedback on each prototype developed iteratively [108]. The current work adapted paper prototyping and quality assurance/ user testing. In addition, they recommended the following design principles for consideration when designing games for CS education [108]:

(i) Define the learning outcomes.

(ii) Research target users.

(iii) Consider challenge and engagement to realise game flow.

(iv) Design negative and positive engagement.

(v) Design appropriate reward systems.

(vi) Design formative and summative assessments as feedback.

(vii) Design for differentiated instructions to cater for differences in learning abilities.

(viii) Provide an opportunity for deliberate practice.

The design process was also guided by communicative and pragmatic design paradigms as adapted from the work by Visscher et al. [269] in the study by Baek et al. [21]. Table

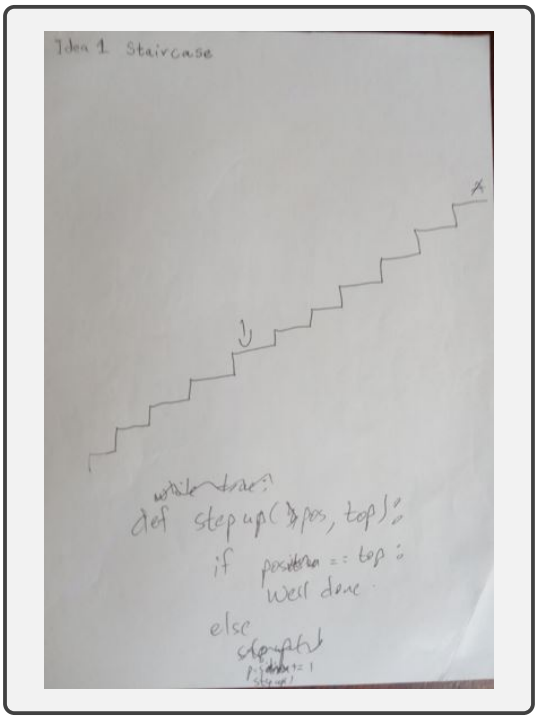


Fig. 4.1 Paper prototype 1

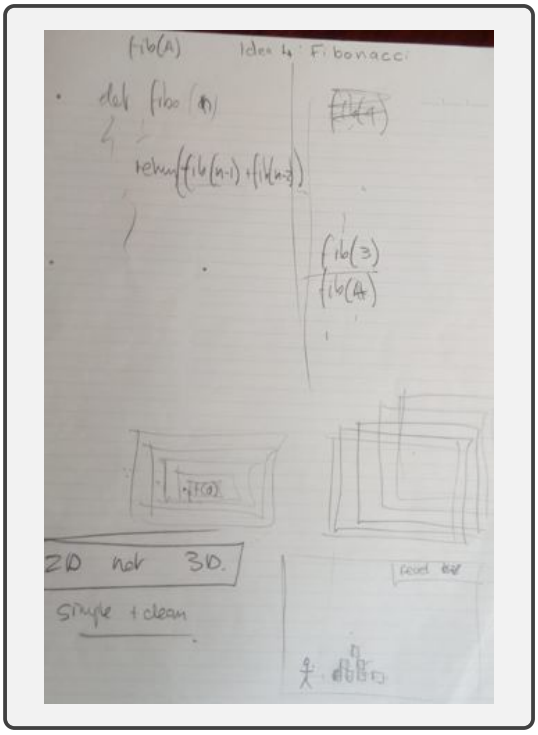


Fig. 4.2 Paper prototype 2

4.1 presents a summary of four philosophy driven instructional design paradigms. Communicative and pragmatic paradigms are among the four. Communicative paradigm was chosen because it : (i) allowed the prototype designer, the researcher and potential users to constantly communicate and reach consensus; (ii) put users at the centre stage of the design process as they played the role of providing information as co-designers; (iii) enabled the prototype designer to play an active facilitation role during prototype dry run sessions. On the other hand, pragmatic paradigm was adopted since it lays emphasis on repeated testing and revision and gives users the role of providing information. Moreover, both paradigms support iterative design process in line with UCD. We adopted three key UCD elements, namely: user participation; contextual inquiry; and iterative design [21].

4.3.1 User participation

Two types of users (primary and secondary) are suggested by Maguire et al. [143]. In this study, CS1 instructors are the primary users (participants) while students are merely secondary users. Their success with the tool largely depends on the extent to which primary users can effectively use it [143].

Table 4.1 Summary of four philosophy driven instructional design paradigms (adapted from [269] in [21])

	Instrumental paradigm	**Communicative paradigm**	**Pragmatic paradigm**	**Artistic paradigm**
Emphasis	Aligns goals and design outcome	Designers and users communicate to reach consensus	Repeated testing and revision	Creative design
Designer's role	Expert responsible for design	Facilitator shares responsibility with users	Expert responsible for design	Artist responsible for design
User's role	Information provider and approver	Information provider and co-designer	Information provider and product user	Product user
Design process	Typically linear	Non linear and iterative	Non linear and iterative	Linear and non linear

4.3.2 Contextual inquiry

This involves considering the users'work needs in context. Context-of-use includes information about users, what they do, physical environment, institutional environment as well as the technical environment [144]. Methods like interviews, discussions, surveys, diaries and field studies have been proposed to collect the required information on the context of use [71]. In this thesis, interviews were used (See Section 3.3). Contextual information such as difficult CS1 topics, time allocated to the topic of recursion by the curriculum, diversity among CS1 learners in the context of Kenya and South Africa, recursion topic teaching scenarios, and instructor game programming technical skills were considered, among others. The list was motivated by the fact that we wanted to design a game authoring platform for CS1 teachers based on their needs, requirements and expectations.

4.3.3 Iterative design

We established early contacts with primary users through a needs assessment study with CS1 lecturers [17]. This was followed by a conceptual qualitative user requirements study [18]. In employing user centered methods to gather design suggestions for learning technologies, Vasalou et al. suggested that users are best involved as informants [267]. During development, we continuously focused on user requirements

and iteratively tested the tool with two experienced volunteer CS1 educators (potential users) from the Department of Computer Science of the University of Cape Town.

Additionally, we sought feedback from postgraduate students in the Information and Communications Technology for Development (ICT4D) laboratory in the department. The total active members of the laboratory at that time was 12. Out of the 12, ten were recruited on purpose. The ten were recruited because: (i) five were CS teachers in their respective home universities, (ii) four had taught CS1 course to first year students previously in the department while two were working as CS1 Teaching Assistants (TAs) at that time, (iii) four belonged to the same departmental research group (Digital Libraries (DL)) as the researcher - hence understood more about the proposed game generator tool, (iv) two were interested in gaming for their research, and (v) another two were researching on Human Computer Interactions (HCI). Table 4.2 presents a summary of the demographic information.

Table 4.2 Demographic information

Variable	**Category**	**Freq**	**Percentage** %
Age	25 - 30 years	3	30
	31 -35 years	5	50
	36 -40 years	2	20
	Above 40 years	0	0
Country	Kenya	2	20
	Uganda	1	10
	Tanzania	1	10
	Malawi	2	20
	South Africa	2	20
	Madagascar	1	10
	Ghana	1	10
Taught CS1	Yes	7	70
	No	3	30
Experience	Less than 5 years	2	28
	5 -10 years	3	44
	Over 15 years	2	28

The two teachers and the ten postgraduate students acted as co-developers by means of cooperation in a non-linear and iterative process [269]. The postgraduate students used the prototype to generate some games. Afterwards, they played the generated games. In addition during student meetings, the researcher presented the prototype and the generated games to the postgraduate students for their feedback on improvements (See Section 4.6.2). In total, three iterative design cycles were conducted. Each yielded a prototype. The prototypes were developed by a contracted Bachelor of Science student majoring in Computer Science and Engineering from the department of Computer

Science. By that time, the student had taken CS1 and Computer games courses. Additionally, he had worked as a CS1 Teaching Assistant (TA). His duties included assisting CS1 students during practical lab sessions and marking tests and practicals. These attributes coupled with his interest in game programming made him suitable for the task. The design and development process lasted for 6 months (December 2019 - May 2020).

4.4 Prototype Development

4.4.1 Review Meeting

During a progress review meeting with the student developer and the two volunteer teachers, the following suggestions were made to make it possible for educators to customise and generate different instances of games:

(i) a built-in template that can allow teachers to easily create custom programming games from given examples.

(ii) game generation steps as in the work by Khenissi [117].

(iii) customisation/ configuration of a game using assets like background color, scene, sprites etc.

(iv) help feature to guide users whenever they perform an illegal operation

(v) an appealing UI.

(vi) download feature for easy distribution of the generated games.

4.4.2 Technologies and the Architecture

Web technologies were used to develop RGG. The application logic contains HTML, JavaScript, PHP, and C#. The pages created from these technologies are designed to run on a Web platform. C# scripts send and access data between the application and back-end server. Cascading Style Sheets (CSS), Java Script (JS), Unity game resources and the database are stored on a Web server. In the front end, the user (instructor) sees and interacts with the system through a Web browser. In the back end, PHP code validates and verifies user requests. During development, the system was tested on a local host. However, for purposes of the intended user studies, the prototype was hosted with Amazon Web Services (AWS). Server side scripting language is used for the prototype authoring platform since it puts less strain on the user's machine.

The prototype generator runs on the server while the generated game runs on the client machine. The Web platform (Web User Interface) allows teachers to easily create/ generate custom games based on their learning content and pedagogical goals. When

generating a programming game instance, a user loads a given game example and uses it to build the game. Additionally, users customise assets such as the ground sprite, wall sprite, background image, and game objects. A built-in game generation template simplifies the game generation process for the non-technical users (educators). The generated standard game is rendered using the Unity engine as an executable file. The generated game is downloaded by the author and distributed to/ shared with students using: (i) google drive (ii) a central database server in a lab; (iii) student laptops; (iv) desktop machines; or (iv) portable storage media. Figure 4.3 is an illustration of the RGG platform's high level architecture.

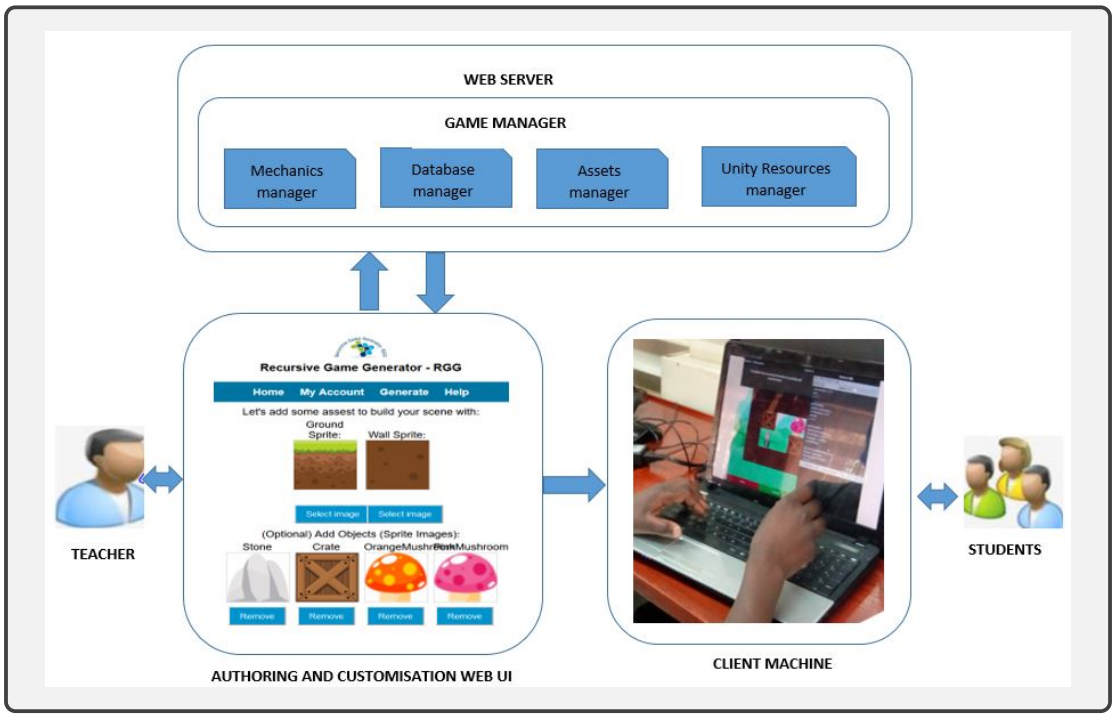


Fig. 4.3 The RGG Architecture

4.4.3 The Prototype Game Generator

In line with recommendations from the team that acted as informants during the development, the prototype game authoring tool has a built-in template that supports both novice and experienced teachers to generate programming games. Figure 4.4 shows the RGG prototype. It also has an appealing interface with a logo. At the same time, it has a login (Figure 4.5) and a help feature (Figure 4.6). It has a download (Fig 4.7) functionality that allows easy download and distribution of the generated games offline. In addition, the prototype generator has a feature to support users with Linux Operating System or Ms Windows.

In order to address the educational games configuration design gaps identified in the reviewed literature, the prototype gives teachers the flexibility to easily configure

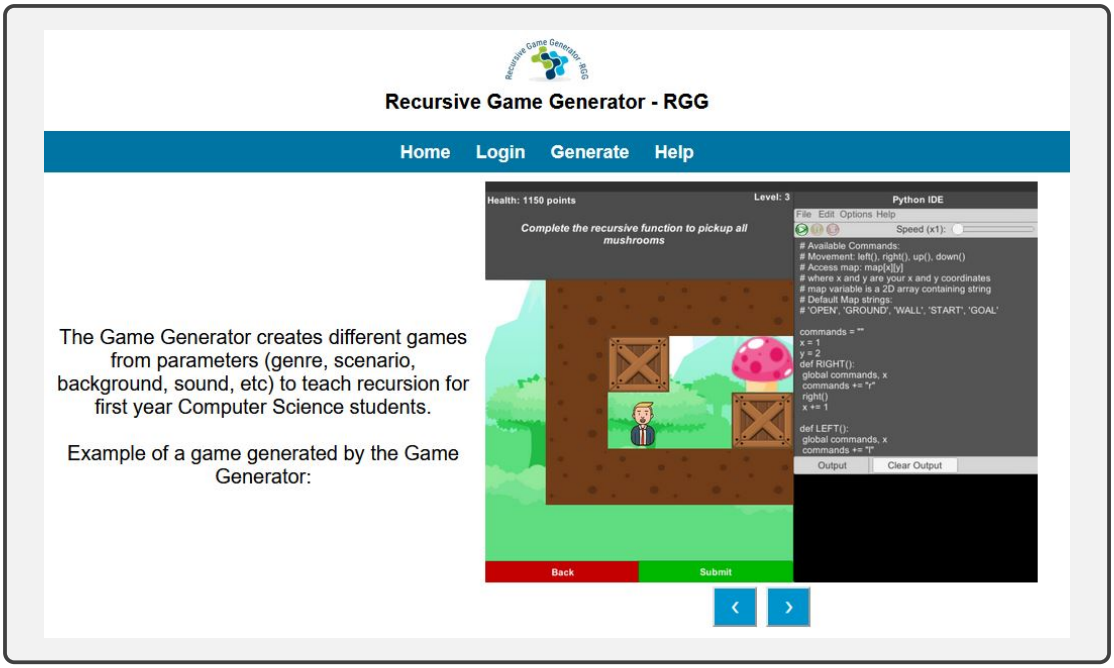


Fig. 4.4 The Recursive Game Generator - RGG

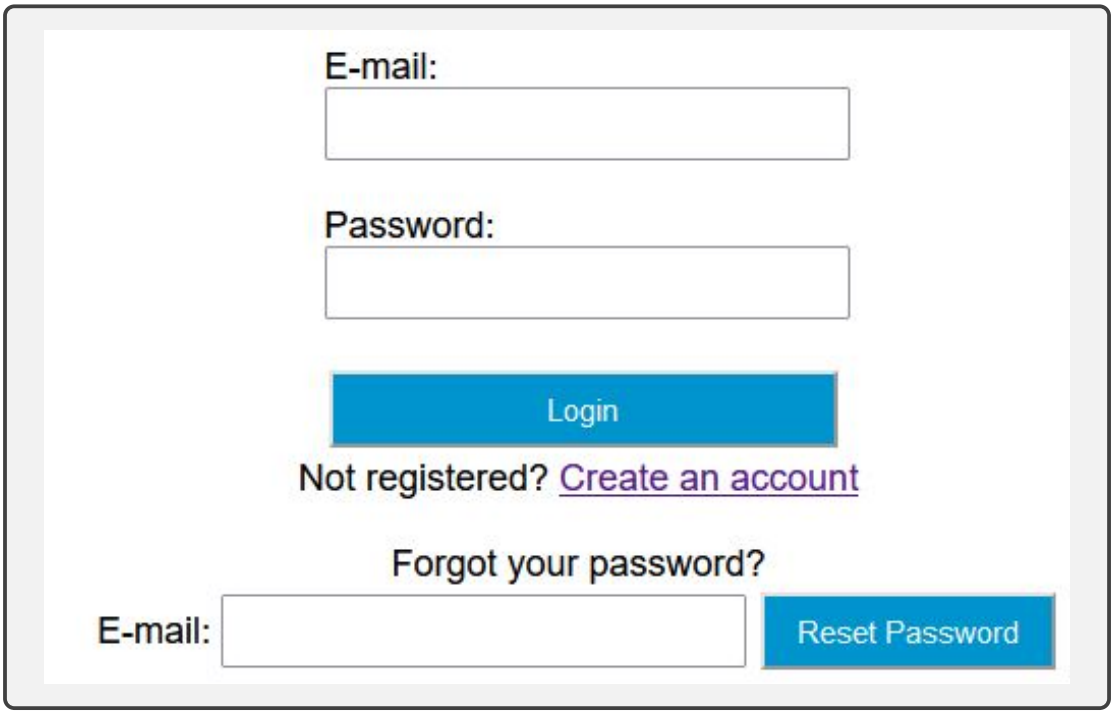


Fig. 4.5 Registration interface

and customise some game parameters such as view, scene, background image, and wall sprites during game generation process. As users progressively get advanced with using the prototype, the design provides an additional functionality that allows for the creation and modification of game content. Consequently, the additional feature enables domain experts (teachers) who want to build a game from scratch to bring in

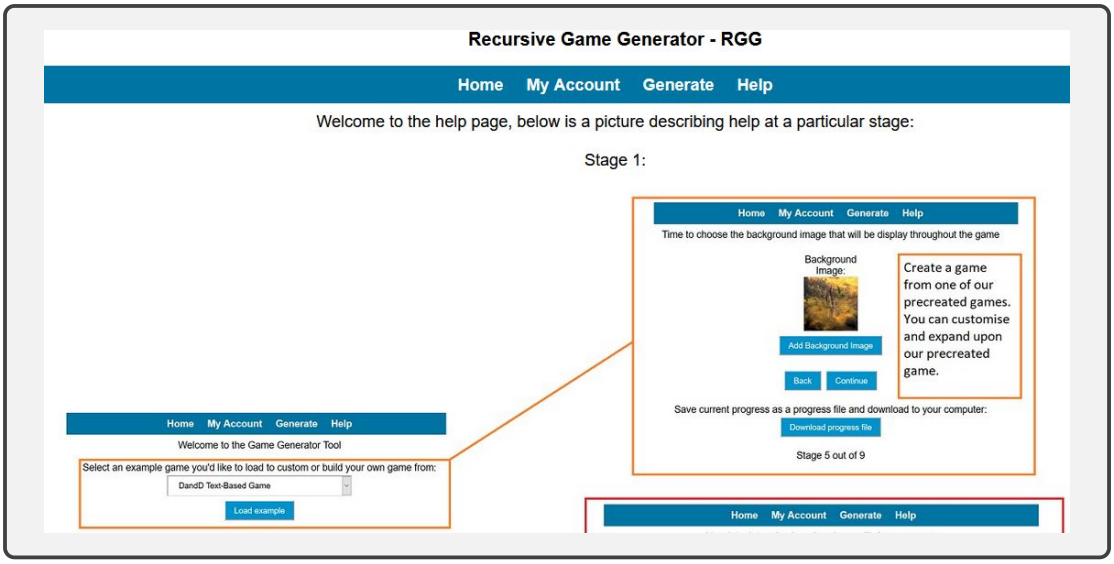


Fig. 4.6 Help Feature

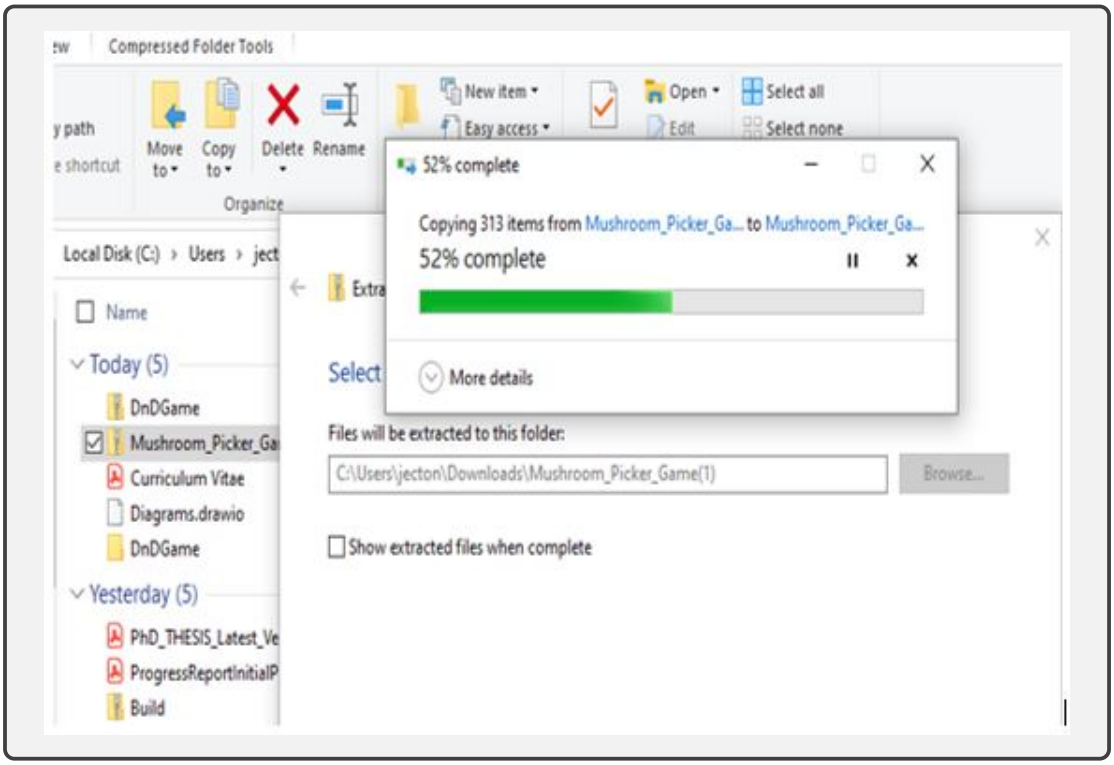


Fig. 4.7 Generated game download

their instructions design speciality [145] into educational games development. Table 4.3 presents a mapping of how Chapter 3 and the design theories were factored in the design and development of the game generator prototype. It summarises the aspects of the game generator's user interfaces and features incorporated.

Table 4.3 A mapping of conceptual design principles, design theories and the designed prototype features

RGG feature	Conceptual design principle	Design theory
Help feature	Provide help facility considering different learning abilities	Complexity theory.
Different game generation interfaces and built-in template feature	Design provides different interfaces for novice and experienced users (Section 4.5 - Step 7)	Differentiating and complexity theories.
Game download feature	Developed prototype allows teachers to download the generated games	Complexity theory
Pedagogical features	Designed and generated games support: (i) different student learning needs (principle 7) and diversity (principles 3 and 4) , (ii) learning by construction, and (iv) visualisation	Differentiating theory
Teaching analogies feature	Designed and generated games support: (i) use of simple mathematical examples (principle 1), and (ii) real life analogies (principle 2).	

4.5 Game generation

When generating a game using RGG, a first time user needs to create an account. This is followed by logging in. Thereafter, game generation is made possible in nine steps.

Step 1 - *Select how you want to create a game*: Once logged in, a teacher can create a game by: (i) using an existing game example, (ii) building one from scratch, and (iii) using a previously saved progress file. In the experiments in this thesis, teachers mostly created games using the given examples.

By selecting the first option, a teacher is presented with three game examples to choose from - the Mushroom Picker, the DnD and the Runner Top Down games (See Appendix C21). Due to time constraints the Runner Top Down game was not fully designed. Consequently, it was not used in the experiments in this thesis. Upon making a choice, a built-in template allows teachers to easily create customised game instances.

Step 2 - *Select how the player will view the game*: If a teacher opts to create a game from scratch, they are prompted to choose how the player will view the game. Three choices are available: (i) text-based, (ii) top-down, and (iii) side view (See Appendix

C22)

Step 3 - *Select the scene you want in the game environment*:
In this step, the users who choose to create a game from scratch are presented with pre-built scenes which they can use to create their games. The scenes are maze, custom map and static game (See Appendix C23).

Step 4 - *Add assets such as ground and wall sprites*:
In this step, teachers select and add game assets such as ground and wall sprites to build the game. In addition, they have the option to add objects like sprite images as well as disable the built-in character (See Appendix C24).

Step 5 - *Select the background image that will be displayed in the game*:
To enable users to create unique game instances, this step allows teachers to select the background image that they want in the game (See Appendix C25). This is significant because images have an effect on the game play environment.

Step 6 - *Add or delete a game level*:
This step enables teachers to add or delete a game level (See Appendix C26). This provides flexibility in the game generation process, enabling teachers to expand existing games, or develop new ones with detailed levels.

Step 7 - *Edit a game level*:
Other than adding a new level, teachers can also edit a game level. At this level, experienced users can create serious games by configuring the: (i) level programming concept, (ii) question, (iii) difficulty, and (iv) code using a built-in template (See Figure 4.8). Otherwise, those who use existing game examples will adopt the concept level, question, difficulty level, and programming code of the chosen game.

Step 8 - *Give the game a title and a brief description* :
In this stage, teachers give a tittle to the level and briefly describe it. These details will be displayed whenever a new game instance is generated (See Appendix C27).

Step 9 - *Download the generated game*: At this stage, the instructor can download the created game (See Appendix C28). This is followed by extracting the downloaded zipped game folder. The folder can then be distributed to students. Upon receiving the zipped folder, students are required to unzip it and run the executable application (CSC500WGameGen) to start playing the game. Other than the executable application, the zipped folder also contains the game generation folder, default assets folder, game

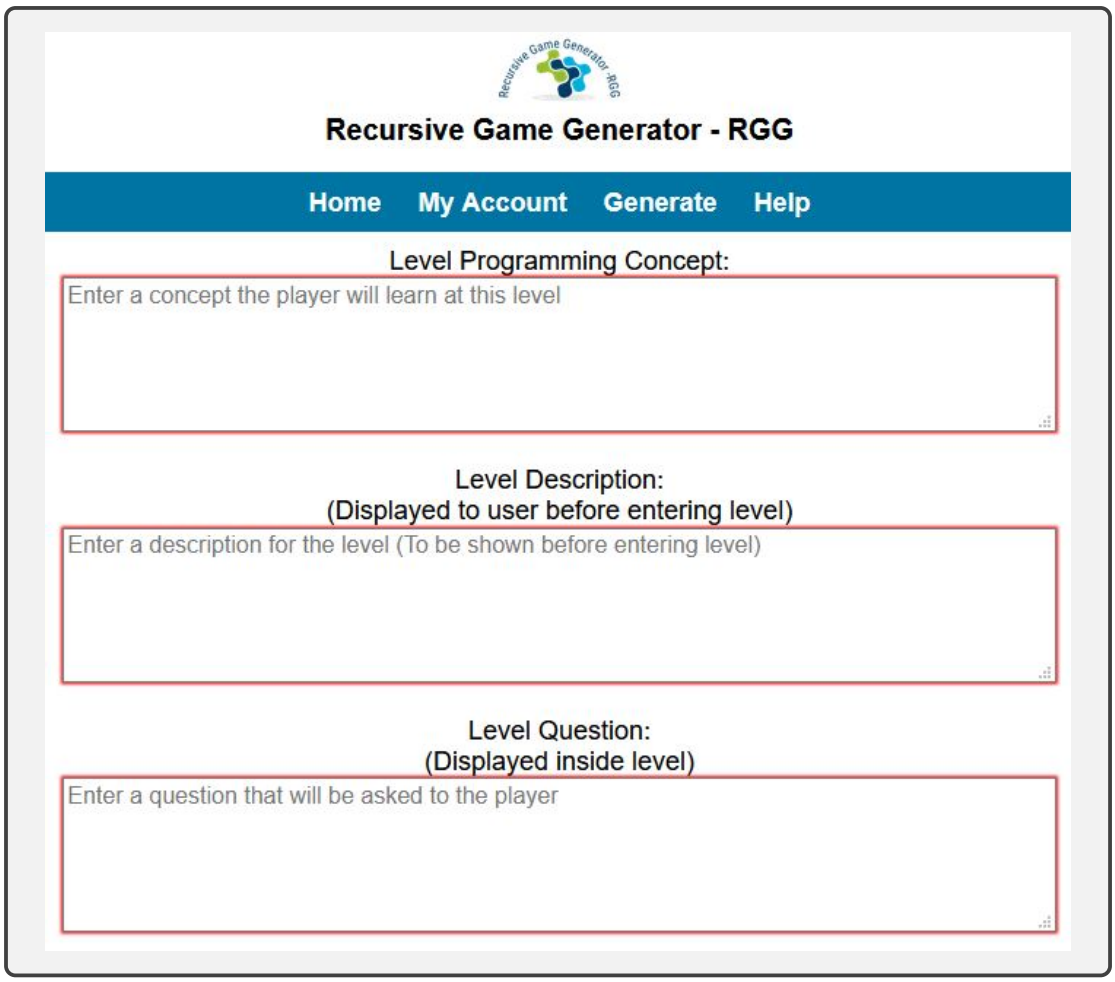


Fig. 4.8 Level concept/ description/ question configuration

resources folder, configuration text file, and Unity applications (Figure 4.9) required to play the game.

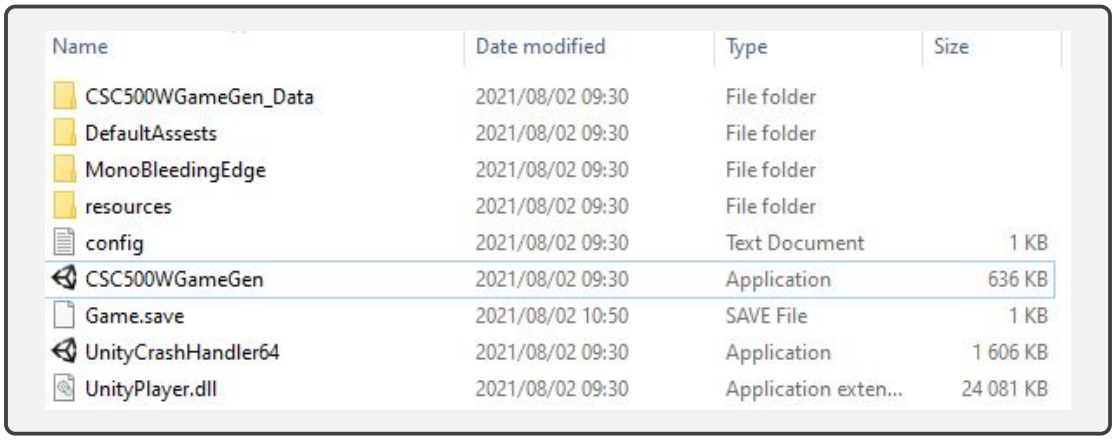


Fig. 4.9 Downloaded Items

Although the prototype generator is designed to allow game generation through 9 steps/ stages, when creating a game using a given example, stages 2, 3, and 7 are

skipped. Likewise, when creating a game from scratch step 4 is skipped. This design approach is meant to simplify the generation process for teachers. The game generation aspects in the skipped steps are automatically derived from the built-in template.

4.6 Generated Games and Features Designed

4.6.1 First Iteration games

Based on the user needs and requirements analysis in Chapter 3, some programming games were designed in the first iteration. This helped to conceptualise the types of games that could possibly be generated by the tool. As an illustration, one of the games was designed based on a real life analogy suggested by one potential user in the first iteration. Figure 4.10 is an illustration of the game.

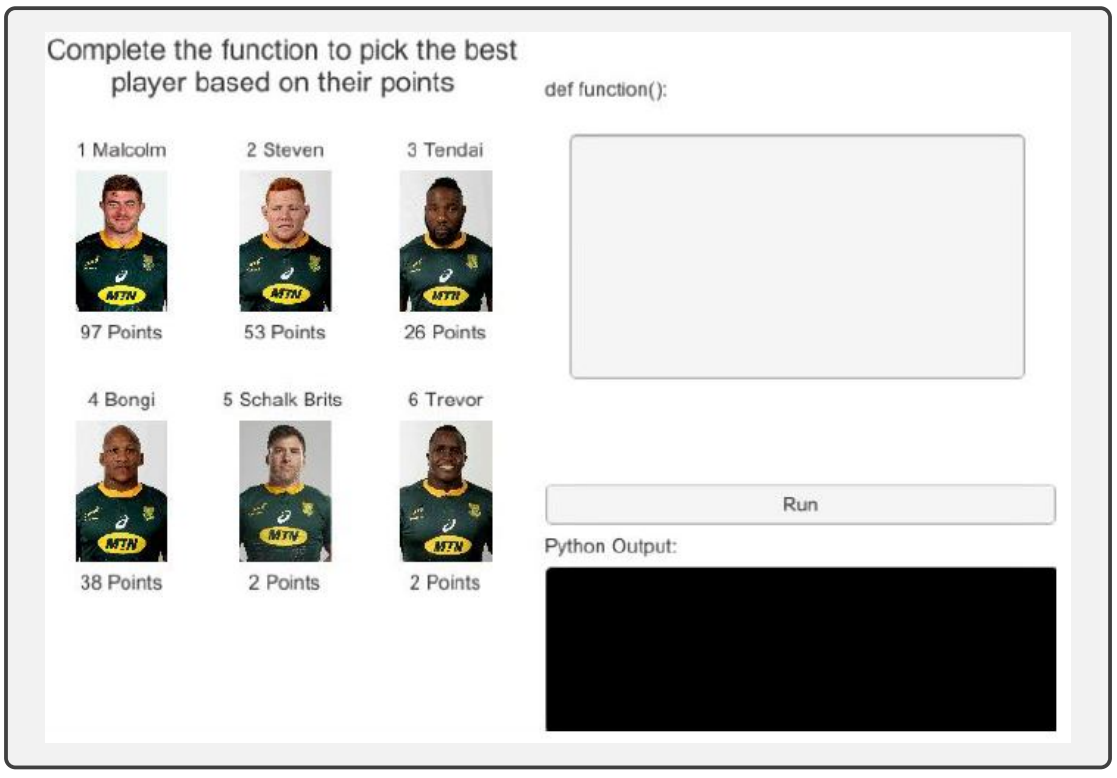


Fig. 4.10 First Iteration Game

Although it was not a fully playable programming game, we were able to idealise some core features. These included pedagogical task, Python script input, Python output, and Team Picker game output. The following functionalities were captured:

(i) a user interface was controllable by keyboard and mouse.

(ii) the interface was kept as simple as possible and as close to an IDE as possible for players to recognise that they can code in Python and get the script output to be able to debug.

(iii) the game could be changed to any different game instance by changing only the player assets.

(iv) the game was a 2D game and was as simple as possible to allow for optimal resource usage.

Some requirements were suggested for improvement of the first iteration games by the researcher and the two volunteer CS1 teachers during a development progress review meeting. During this meeting, the student who had been contracted to develop the prototype presented the games that had been designed. Some of the recommendations that were made for consideration in the second and third iteration games were:

(i) the games should be fully playable.

(ii) the games should have at least three levels of increasing difficulty.

(iii) the games should have a point system to motivate players.

(iv) the games should have real life scenarios and simple mathematical examples used by most teachers. when teaching the recursion topic such as the factorial of a number, power of a number, binary search etc.

(v) the games should support visualisation through character movement in a recursive manner as in the work by Chaffin et al. [48].

4.6.2 Second and Third Iteration Games

Most design suggestions from the first iteration were incorporated in the games designed during the second and third iteration. Three programming games were designed. Two were successfully completed but one was not. The two were the Mushroom Picker and the DnD games. Each game had at least three levels with increasing difficulty.

To further improve the generated games and the prototype generator tool, the ten postgraduates participants actively participated in the design of the second and third iteration games. This was done during meetings organised by the representative of postgraduate students. Supervisors do not attend such meetings. Three meetings were held. The researcher presented the prototype and generated games to the participants in the Gary Marsden meeting room. This was followed by suggestion for improvement. Some of the suggestions were:

(i) the games should allow students to progress gradually from easy to difficult levels.

(ii) students/ players should be able to see their progress as they solve the programming tasks.

(iii) from the HCI perspective, the tool and the generated games should be simple, visually appealing, efficient and usable.

(iv) the generated games should be easy to learn and remember.

(v) the generated games should provide immediate feedback to the player.

(vi) the students should be able to visualise execution of recursive functions as they play the generated games.

The proposed suggestions were given to the student developer for consideration in the second and third iteration games and prototype design for improvement. Figure 4.11 and Figure 4.12 are illustrations of the DnD and Mushroom picker game instances respectively.

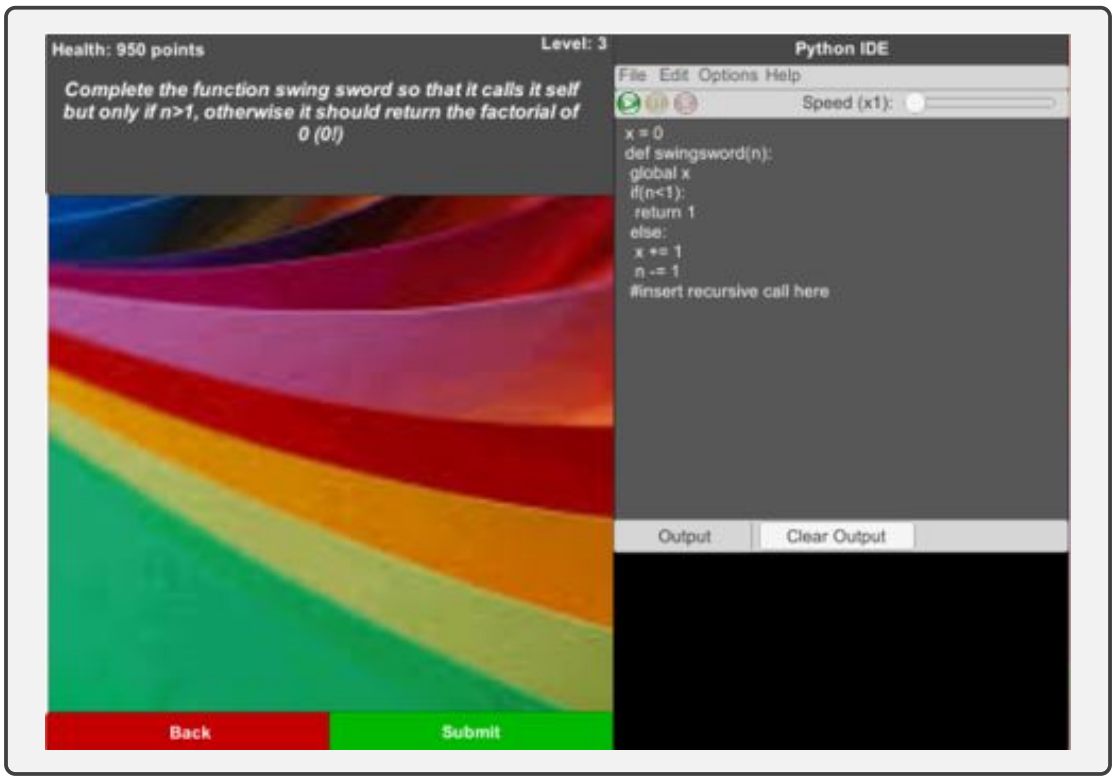


Fig. 4.11 DnD game instance

4.6.3 Features of the Generated Games

Pedagogical goal: It is worth noting that the games generated by the prototype in the third iteration present players (students) with a pedagogical goal/ learning task (Figure 4.14). This feature appears above the visualisation feature. Tables 4.4 and 4.5 demonstrate the alignment of the Mushroom Picker and DnD game levels to pedagogy [126, 148] in CS1.

Table 4.4 Alignment of the Mushroom picker game levels to pedagogy in CS1

Level	Learning outcome/ task	Programming concept	Python code snippet

1	Complete the given code to pick up all mushrooms in the level	Function	`def pickUpAllMushrooms():`
			`#insert commands here`
			`#available commands:`
			`RIGHT()`
			`LEFT()`
			`UP()`
			`DOWN()`
			`PICKUP()`

2	Complete the given code snippet by entering a stopping condition for the algorithm	Recursion - Stopping condition	`def pickUpAllMushrooms():`
			`#insert stopping condition below`
			`if():`
			`return:`
			`RIGHT()`
			`PICKUP()`
			`RIGHT()`
			`pickUpAllMushrooms()`

3	Complete the recursive function to pick up all the mushrooms	Full Recursion	`def pickUpAllMushrooms():` `#insert code here`

Table 4.5 Alignment of the DnD game levels to pedagogy in CS1

Level	Learning outcome/ task	Programming concept	Python code snippet
	Complete the given code so that the function returns double the given parameter x	Function	`def double(x):` `#insert code here`
2	Complete the given code so that the swingsword function returns the factorial of 0 if the parameter n is smaller than 1 otherwise it returns 0	Recursion - parameters, base case, recursive call	`def swing sword(n):` `#insert code here`

3	Complete the function swingsword so that it calls itself but only if n is greater than 1, otherwise it should return the factorial of 0	Recursive call with stop condition	`x=0`
			`def swingsword(n):`
			`global x`
			`if(n<1):`
			`  return 1`
			`else`
			`  x += 1`
			`  n -= 1`
			`#insert recursive call here`

4	Complete the code snippet so that the function swingsword returns the factorial of the parameter n given to it (n)	Recursive call	`recursiveCalls=0`
			`def swingsword(n):`
			`  global recursiveCalls`
			`  recursiveCalls += 1`
			`#insert code here`

Fig. 4.12 Mushroom Picker game instance

Integrated Development Environment: The generated games have an IDE feature that is useful for monitoring student problem solving activities and coding behaviour [140]. This feature allows learners to compete python code snippets and run their algorithms [101, 68, 126, 148]. Any syntax or logical errors made by a student are displayed in the output area. Figure 4.14 shows the feature. Furthermore, the Mushroom picker game allows students to use simple commands such as UP(), DOWN(), LEFT(), RIGHT()[2, 56], which make programming easy for novices.

Visualisation: There is growing interest in the use of visualisation as an effective pedagogical strategy for teaching abstract concepts to CS students [186, 197, 278]. This design approach has the potential of engaging learners in an active and interactive environment [66]. One of the key design features of the games generated by the proposed tool in the second and third iterations is an interactive visual environment that allows learners to view the execution of their code in a recursive manner through character movement [48, 56]. Figure 4.14 illustrates a visualisation of level 3 game play interface of a sample game built from the Mushroom picker game. Figure 4.13 is an illustration of an instance of a game created from the Mushroom picker game example.

Fig. 4.13 Mushroom picker game Level 3 play interface

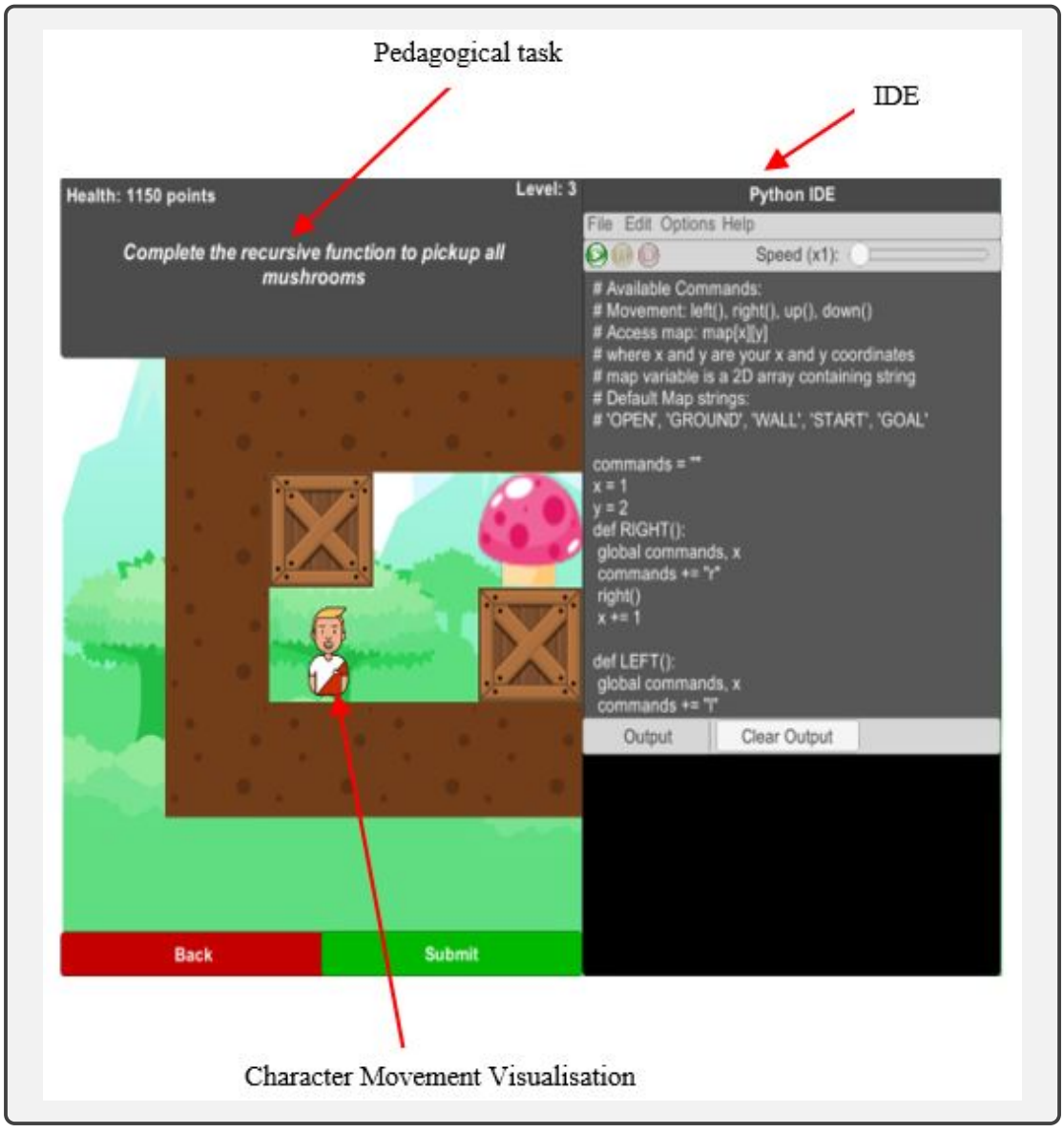


Fig. 4.14 Visualisation, IDE and Pedagogy features

Motivation/ reward: In CSE, some works have underscored the motivational impact of serious games on CS1 students [27, 229, 246]. Consequently, motivation is another important feature that was considered during the games design. For each level completed successfully, the generated games award trophies and health points to learners as recommended by some serious games design scholars [68, 101, 148, 126]. This provides instant reward [209] to effectively optimise such engagement, hence reinforces learning. This is shown in Appendix C29.

4.7 Summary

This Chapter presented the User Centered Design approach adopted to design and develop the Recursive Game Generator Tool prototype. UCD is a common software development method that lays emphasis on user participation, contextual inquiry, and iterative design. The iterative nature of these phases gave the author an opportunity to continuously refine the requirements and the prototype with potential users. In addition, unique design features such as configuration/ customisation, pedagogical goal, IDE, visualisation, and motivation/ reward guided the design of the experimental platform evaluated in the next Chapter.

CHAPTER 5

Experimental Design

This Chapter presents the evaluation protocol adopted for this study. The purpose of the study was to investigate the conceptual design attributes and the effectiveness of a game generator tool to support CS educators to teach programming. In addition, the educational potential of the generated games was investigated. Ainsworth and Fleming contend that the success of any learning material authoring tool depend on two factors: (i) *"the extent to which the authoring tool is usable by its intended user population (classroom teachers, university lecturers, adult trainers)"*, and (ii) *"whether the resulting systems are effective at supporting learning"* [5, p. 1]. Given its difficulty among students, the recursion topic was used as a case study. User needs and requirements analysis studies [17, 18] informed the design and development of a prototype game generator tool called RGG. In order to evaluate the concept of a game generator tool in higher education, four user studies were conducted. Three were done with the generator and one with the generated games. CS1 programming teachers evaluated the prototype being the primary users while students evaluated the generated games as the secondary users. Teachers used the prototype to create custom games and answered the USE questionnaire [135] based on their experiences. Additionally, they completed the Attrakdiff questionnaire [94]. On the other hand, CS1 students played the generated games and answered the Game Experience Questionnaire GEQ [103].

Findings of the first, second and third experiments provided empirical evidence that helped in answering the second research question: *How effective is the use of a generator tool by CS educators in creating games to teach recursion*? On the other hand, findings of the fourth experiment provided empirical evidence that helped in answering the third research question: *What is the suitability and educational potential of the generated games to enable students to learn the topic of recursion*? The reasoning behind the three experiments conducted with the teachers is that educators are the target primary users of the proposed game generator tool, students are merely secondary users. The success of the students depends largely on the ability of teachers to effectively use the tool to create games. Experiment four was done with CS1 students as potential secondary users of the tool. By having students play and evaluate the generated games, their educational suitability and potential was demonstrated. This validated the

games. The protocols followed for experiment one, two, three and four are presented in Sections 5.1, 5.2, 5.3 and 5.4 respectively.

5.1 Experiment One

5.1.1 Purpose

Some scholars have reported that authoring tools ought to be usable by practicing classroom teachers [180]. In line with this, this was an online experiment designed to evaluate the usability of a prototype game generator tool called RGG designed to support CS educators in higher education to create games to teach introductory programming. The basic idea is that using RGG the instructors, even without game programming skills, could easily create serious games for CS1.

5.1.2 Participants

CS1 instructors from higher learning institutions from Kenya and South Africa participated in this evaluation study. Kenya and South Africa were chosen mostly for convenience purposes. The subjects from South Africa were recruited from a list of 23 programming lecturers created during the Southern African Computer Lecturer's Association (SACLA) conference in 2018. The researcher, who was in attendance, explained the intention of the study to the lecturers during tea and lunch breaks. This was followed by a request to participate. Lecturers who were willing to take part shared their contact details with the researcher. Afterwards, an invitation email was sent out. On the other hand, respondents from Kenya were recruited by snowball sampling through recommendations from colleagues since the researcher (a programming lecturer from Kenya) had already established a few contacts.

5.1.3 Tasks

Participants performed the following tasks:

(i) create user account and log in;

(ii) complete a demographic questionnaire;

(iii) create a custom game using a given DnD and Mushroom Picker game example;

(iv) customise the created DnD and Mushroom Picker games;

(v) download the created DnD and Mushroom Picker games;

(vi) play test the created DnD and Mushroom Picker games; and

(vii) complete the post-test questionnaire and rate:

- task performance experience
- tool ease of use
- tool learnability
- tool usefulness
- satisfaction with tool
- overall opinion about the tool

5.1.4 Procedure

Ethical clearance was obtained from the University of Cape Town. A pilot study was first conducted with five volunteer instructors from the University of Cape Town. The intention was to (i) ensure that respondents understood the questionnaire terminology, (ii) reduce chances of bias due to leading questions, and (iii) ensure that the questionnaire could be completed in a reasonable amount of time. Afterwards, invitation emails were sent to potential participants for the main study. Willing participants signed up and completed a consent form. This was followed by sending an online pre-test survey.

Upon receiving a completed pre-test survey from each participant, an online post-test survey was sent out. The post-test survey was sent only to respondents who completed and submitted the pre-test survey. In addition, a link to the prototype game generator tool, a video tutorial and a task sheet were provided. The design approach adopted reduced chances of social desirability bias [185] and gave users a reasonable exposure to the tool (one month). This reduced chances of variability in the usefulness dimension ratings [135]. Upon completing the post-test survey, interested participants entered into a draw for a chance to win 500 South Africa Rand as compensation for their time and internet data. In total, ten prizes were given. A summary of the procedure follows:

5.1.5 Data collection Methods

Quantitative UX data was collected through user self-reporting. Likert scales were used to collect both pre-test and post-test study data. The standard USE questionnaire [135] was adopted, given the recent evidence of its reliability and validity [122]. The pre-test survey contained questions about demographics. On the other hand, the post-test survey had questions on the user task experience, RGG's usability and one last overall question on final comments about the tool. Data on the final user comments was collected through an open ended question. All responses were collected and stored using the LimeSurvey tool. This ensured reliability of the given answers as each participant only answered once using a given token.

5.1.6 Criteria Used to Answer Research Question 2

RQ 2. *How effective is the use of a generator tool by CS educators in creating games to teach recursion?*
In an attempt to answer this research question, experiment one sought to provide evidence of the effectiveness of the prototype by understanding the user experiences of CS1 programming instructors from Kenya and South Africa when using it as a means of creating serious games [16]. The following four sub-questions were answered:

1. *What is the post task experience of CS1 instructors from Kenya and South Africa after using the RGG prototype game generator tool*?

2. *How useful and easy to use do programming instructors from Kenya and South Africa find the RGG prototype game generator tool*?

3. *How do programming instructors from Kenya and South Africa find the learnability of the RGG prototype game generator tool*?

4. *How satisfied are programming instructors from Kenya and South Africa with the RGG prototype game generator tool*?

To address the four sub-questions, this experiment measured perceived task performance metric and the prototype's usability. In user experience studies, *"a metric is used as an indicator that is measured to gauge the experience of the user while interacting with a product, system or service"* [104, p. 3]. Four metrics have been predominantly used in the UX domain, namely: (i) task performance metric, (ii) issue-based metric, (iii) self-reported metric, and (iv) behavioural and physiological metric [8]. Task performance metric evaluates the experience of users while interacting with the system. Issue based metric assesses the issues or problems that users encounter while using the system whereas self-reported metric evaluates users'opinions about a product or system after using it. Physiological metric measures how users actually feel when using a system. In this online experiment, we used task performance and self reporting metrics. Task performance metric was measured based on three criteria: (i) task difficulty [8], (ii) task success [8, 100], and (iii) the estimated time on task [8, 100]. Though recommended as a complementary measure of UX [279], we did not use physiological metric such as galvanic skin response given the online nature of the experiment.

5.1.7 Task difficulty

To understand the level of difficulty users experienced when performing different tasks, participants were asked to rate the extent to which they agreed or disagreed with the statements on task easiness. Histograms were then used to present frequencies.

5.1.7.1 Task success

To evaluate the effectiveness of the tool in supporting CS1 educators when using it to create games to teach programming, respondents were asked to perform the tasks and rate their levels of success. Task success was measured on three levels: complete success, partial success and failure. Complete success and partial success were further broken down into success with or without assistance. Assistance involved seeking help from the help menu. Failure was further categorised into two: (i) thought the task was easy but it was not, and (ii) failure - gave up (meaning the participant failed at performing the task and just gave up). These levels were clearly explained to participants before using the tool [8].

5.1.7.2 Estimated time on task

To measure the tool's efficiency, participants were asked to give an estimate of the time they spent while performing each of the four tasks. This was done on a 5 point scale. Time ranges (discrete time intervals) were used, where 1 represented 10 to 13 minutes. 2 represented 7 to 10 minutes, 3 represented 5 to 7 minutes, 4 represented 3 to 5 minutes and 5 represented 0 to 3 minutes. Like the task success, histograms were used to present frequencies.

Meanwhile, usability was evaluated based on four metrics: (i) usefulness; (ii) ease of use; (iii) ease of learning and (iii) satisfaction. These were measured through the standard USE questionnaire [117]. USE was adopted because of the demonstrated evidence of its reliability and validity [122].

In literature, Technology Acceptance Model (TAM) [59] assumes that acceptance of any technology depends on the perceived usefulness, and ease of use. Some researchers also found out that teachers's perceived usefulness and ease of use of technology play a key role in determining their intention to use it [232]. Therefore, in this experiment, participants evaluated the prototype's usability [190] through self-reported data using four metrics, namely: perceived (i) usefulness; (ii) ease of use; (iii) learnability; and (iv) satisfaction with it.

5.1.8 Analysis and Presentation

Statistical Package for Social Sciences (SPSS) software was used to perform descriptive analysis on collected data. USE was then scored separately for each of the four dimensions (usefulness, ease of use, ease of learning, and user satisfaction) by calculating the percentages of respondents in each category of the Likert scale for each item in the dimension. Stacked bar charts are used to analyse and present the percentage of users who fall into each category or level [8]. These percentages are then compared

across the categories or tasks. For the estimated time on task and perceived task success, histograms are used to present frequencies. On the other hand, self-reported text responses from final user comments are categorised, coded and analysed thematically [273]. Emerging themes were discussed and agreed upon by the researcher and the supervisor. Only results from respondents who completed both the pretest and post-test questionnaires are included in the final analysis.

5.2 Experiment Two

Some scholars have also argued that the adoption potential [22, 23] of the GBL approach generally depends on its acceptance by classroom teachers [40, 95]. Luxton et al. underscored the need to test CS1 support tools with other instructors within the wider CSE community [139]. Therefore, to further demonstrate the usability and adoption potential of the proposed game generation idea, we conducted a second user study with CS1 educators drawn from the wider CSE community.

5.2.1 Purpose

This experiment was conducted to gain further insights from the wider CSE community about the usability of the prototype based on additional usefulness items on top of the ones provided by the USE questionnaire. Additionally, the aim was to explore specific design features of the prototype. This study was considered useful because its findings would provide insights on what features may be particularly well-suited to programming games development.

5.2.2 Participants

We recruited CS1 higher education instructors from the CSE community as participants given that the statements and questions created mainly reflected practices and challenges related to their work. To reach this target group, we sent the survey invitation to two mailing lists. These were the Southern African Computer Lecturers'Association (SACLA) mailing list and the SIGCSE-members listserv. Some participants were recruited by reference and introduction by colleague lecturers. In addition, some of the respondents were recruited during one of the ACM sponsored conferences where the researcher was a participant. In total, this experiment had 28 participants. It is worth noting that this was a follow up of experiment one. Potential participants from Kenya and South Africa who had participated in the first experiment did not take part in the second study.

5.2.3 Procedure

The following procedure was followed:

(i) ethical approval was obtained from the University of Cape Town;

(ii) invitations were sent to SACLA, SIGCSE-members mailing lists and other potential participants'email addresses;

(iii) links to the prototype, a video tutorial and a task sheet were provided in the survey;

(iv) willing participants completed an online consent form;

(v) participants used the prototype to perform the experiment tasks (See Sub-Section 5.1.3); and

(vi) participants completed an online survey.

5.2.4 Criteria Used to Answer Research Question 2

RQ 2. *How effective is the use of a generator tool by CS educators in creating games to teach recursion?*

To answer the research question 2, this experiment addressed two sub-questions:

1. *How useful do CS1 instructors find the prototype game authoring tool after using it to create games to teach recursion?*

2. *What design features do CS1 instructors find most useful?*

To answer the first sub-question, we tested additional usefulness items other than the ones proposed in the standard USE questionnaire. To answer the second sub-question, we investigated the usefulness of some unique prototype design features. The features included (i) custom game generation, (ii) built-in template, (iii) visualisation, (iv) support for Windows and Linux Operating systems, (v) Python IDE, (vi) no game programming required, (vii) Help feature, and (viii) game download and distribution support, among others. Teachers provided feedback on the design features they found most useful. This data was considered useful as it provided an assessment of particular tool features. The finding could inform design of future programming game authoring tools.

5.2.5 *Data collection and Analysis*

Quantitative UX data was collected through user self-reporting. The first section of the survey had questions that collected demographic information. The next section elicited data concerning the perceived usefulness of the prototype. Another section had a question that sought to understand the prototype's features that were perceived by instructors as most useful. All collected data was stored in Lime Survey. For analysis, we used SPSS software and descriptive statistics. Thematic content analysis was used to analyse qualitative responses. Open-ended responses were transcribed using Microsoft Word software. Each free text-response was coded with one or more categories. The researcher and the supervisor independently studied the responses to come up with the categories. Where there were disagreements, meetings were set to harmonise them. Tables are used to summarise and present findings.

5.3 Experiment Three

In this experiment, the prototype game generator tool was empirically evaluated in a controlled laboratory user experiment with trainee teachers [16]. A within-subject design was employed. Participants were asked to evaluate UX when creating custom games using RGG. The games were created from two game examples (the DnD game and Mushroom picker game) that were given to users. The research team comprised of two people. They included the main researcher and a volunteer research assistant. The research assistant was a CS high school teacher from Kenya who had interacted with the prototype in the first experiment.

5.3.1 *Purpose*

The controlled experiment sought to investigate the subjective usability and user experiences (dependent variables) of CS trainee teachers when using a prototype game generator tool to create games to teach the recursion topic. It was also meant to test the adoption potential by this group of potential users.

5.3.2 *Participants*

The participants of the study were CS trainee teachers from the Faculty of Education. This target group lacks practical pedagogical experience. Moreover, like CS Teaching Assistants (TAs), they are likely to face teaching challenges [218, 217] and probably would have limited classroom management skills [136]. Consequently, it was believed that they could benefit more from the proposed idea of a game generator tool. This could particularly be the case during their teaching practice or immediately when they

get employed. Indeed, a recent study underscored the need to seek the views of teachers, including pre-service and in-service trainees whenever new tools and methodologies are considered in the teaching and learning process [109]. The authors argue that this is critical since teachers are the ultimate adopters of such strategies and tools in practice. Their attitudes and views could influence the intention to use the tools in future. In this regard, the subjects of this experiment were training as high school GBL teachers. Therefore, one of their teaching subjects was computing. Participants were drawn from Kenyatta University (KU) in Kenya (a University known for training high school teachers). The target respondent group was motivated by the fact that (i) at the time of the experiment, most universities in the country have resumed face to face learning after eight months of closure due to the Covid-19 pandemic and, (ii) the ethical clearance process is faster in Kenya than in South Africa. In addition, it was assumed that the target respondents (i) had been taught some programming courses, (ii) had learnt how to plan teaching computing in high school, and (iii) would definitely encounter the challenges of teaching programming (especially the difficult topic of recursion).

5.3.3 Materials

The following materials were used in the experiment:

(i) an online informed consent form;

(ii) a print out of task sheet;

(iii) a video tutorial;

(iv) a computer with a Web browser and Internet connection;

(v) the Attrakdiff 2 questionnaire; and

(vi) an exit survey.

5.3.4 Tasks

Tasks performed by participants included:

(i) log in using a given user account and password;

(ii) create custom game by customising assets;

(iii) download the generated game;

(iv) customise the generated game;

(v) play test the generated game;

(vi) complete the Attrakdiff 2 questionnaire; and

(vii) complete the exit survey.

5.3.5 Criteria Used to Respond to Research Question 2

RQ 2. *How effective is the use of a generator tool by CS educators in creating games to teach recursion?*
To answer the second research question, this experiment sought to address 2 sub-questions:

1. *What is the adoption potential of a support tool that uses the concept of a game generator to help CS trainee teachers to create games that can teach programming?*

2. *What is the overall opinion of CS trainee teachers about the RGG and the generated games?*

Game generation and play experiences were measured through subjective ratings of the (i) usability, (ii) user experience, and (iii) qualitative statements. The summative measures were taken from a suitable sample of potential users who performed given tasks in a realistic context of use [36]. The AttrakDiff questionnaire developed by Hassenzahl et al. [94] was used to assess the perceived pragmatic quality, hedonic quality, and attractiveness of the prototype game generator. The questionnaire has four scales with a total of 26 items. The items that are on a scale from -3 to +3 are measured as a semantic differential. Each item consists of a pair of terms with opposite meanings.

5.3.6 Procedure

Ethical clearance and study planning
Ethical clearance was first sought from the University of Cape Town. This was followed by acquiring two more research approvals, namely a research permit from the National Commission for Science, Technology and Innovation (NACOSTI) in Kenya, and an approval letter to conduct research at Kenyatta University in Kenya. The researcher established rapport with two lecturers who acted as contact persons at Kenyatta University. One was from the Faculty of Education while the other came from the Department of Computer Science.

During every visit to the University, the research team strictly adhered to the protocols laid down to mitigate the spread of the Covid-19 pandemic. Some of these included (i) identification and registration with the Campus Security Officers (CSOs) manning

the main entrance gate, (ii) frequently washing and sanitising hands, (iii) wearing face masks, and (iv) keeping social distancing. At the Department of Computer Science, the research team held two meetings with the contact lecturer and the class representative to plan the study. The lecturer had just concluded practical programming classes with the learners in a CS1 course. The lecturer had taught basic programming concepts such as variables, loops, conditionals, procedures, functions and lists. The concept of recursion had also been introduced.

A follow up meeting was later scheduled with the target participants in the laboratory. During the meeting, the lecturer introduced the research team. Before each meeting, all participants washed their hands using the facility at the entrance of the department's building and sanitised inside the laboratory. The team was neutral to the participants as none of the members was a lecturer at the University. The team then explained the purpose of the study and demonstrated the way in which the prototype game generator tool works, using a video. This was followed by requesting the trainee teachers to participate in a study to evaluate the prototype. Through the class representative, the researcher announced the study in the class mailing list and other social media platforms. A week before the main experiment, two further visits were made to the department to secure and set up the laboratory.

Pilot study

First, a pilot study was conducted with five out of the 27 volunteer participants. The five did not participate in the main experiment. They used the prototype to create custom games from the DnD game example and completed the Attrakdiff 2 questionnaire. The aim was to ensure that the (i) Lime Survey platform and Amazon servers (on which the prototype was hosted) were stable, (ii) respondents understood the Attrakdiff 2 questionnaire terminology, and (iii) questionnaire could be completed in a reasonable amount of time.

Main experiment

A laboratory with internet access was set up for the main experiment. The same laboratory used to teach the students during practicals was used. This gave respondents a familiar environment [125]. A total of 22 participants who did not participate in the pilot study were divided into two equal groups (11 students each). Each participant was randomly assigned to one of the groups. Before the experiment, the team ensured that each respondent wore a face mask and sanitised. A printout of the task sheet was then given to each participant. After that, the research team again explained the purpose of the experiment and the tasks to be performed. Each participant signed an online informed consent form. The first group (Figure 5.1) started by using the prototype to create custom serious games from the DnD game example, then played the generated

games. This was followed by completing an online Attrakdiff 2 questionnaire. Next, the group generated serious custom games from the Mushroom Picker game example. Afterwards, they played the generated games, then again completed an online Attrakdiff 2 questionnaire. Finally, every group one member completed an online exit survey comprising demographic information and wot optional open-ended questions. These questions allowed the users to further express any personal general views about the prototype and any negative experiences for future improvements. During the experiment, other than the participants, only the researcher and the research assistant were present. Their roles included clarifying the tasks to participants and answering queries. Before group one participants left the laboratory, the researcher thanked them for their time.

This procedure was repeated for the second group (Figure 5.2) until all the 22 participants completed evaluating the prototype. However, for the second group, the order was reversed. Participants first generated custom games from the Mushroom picker game example followed by from the DnD game. The two groups did not interact with each other during or immediately after the first experimental laboratory session. Each test session lasted for approximately one hour and 40 minutes. The following is a summary of the procedure followed during each laboratory session:

Fig. 5.1 Group 1 Participants

Fig. 5.2 Group 2 participants

(i) The team ensured that each respondent wore a face mask and had sanitised;

(ii) the participants were welcomed to the experiment;

(iii) a total of 22 participants were randomly assigned to 2 groups, each comprising 11 students;

(iv) the first group remained in the laboratory to do the experiment;

(v) invitation to the survey was sent to the participants;

(vi) participants signed an on line informed consent form;

(vii) an online link to the RGG prototype was given to participants;

(viii) participants were issued with a printout of the task sheet;

(ix) participants were issued with a print out of the Attrakdiff questionnaire;

(x) the research team explained the purpose of the experiment and the tasks to be performed;

(xi) participants were taken through the meaning of the opposite word pairs of the Attrakdiff 2 questionnaire;

(xii) participants used the prototype to create custom serious games from the given game examples;

(xiii) participants play tested the generated games;

(xiv) participants completed an online Attrakdiff 2 questionnaire and an online exit survey;

(xv) participants returned the task sheet and the Attrakdiff questionnaire print outs; and

(xvi) the researcher thanked participants for their time.

(xvii) Steps five (v) to sixteen (xvi) were repeated for the group 2 participants.

5.3.7 Data collection and analysis

Both quantitative and qualitative UX data were collected through user self-reporting. Three surveys were answered by each respondent. The first and the second used Likert scales to collect overall subjective user experience using the standard Attrakdiff 2 questionnaire. The third was an exit survey containing demographic information and two open ended questions. The first question was about the most negative issues while the second captured overall comments about the prototype and the game generation idea. We used SPSS to compute and analyse the mean score for each scale of Attrakdiff. Mean values of the word pairs were computed and analysed. The mean values were compared across the pragmatic quality, the hedonic quality, and the attractiveness scales. The evaluation was pegged on a 7-point semantic differential scale (i.e.-3: complicated, +3: simple). For analysis, we gave answers the values:-3, -2, -1, 0, 1, 2 or 3. Text responses from the final user comments were categorised, coded and analysed thematically [273]. NVivo 12 software was used to analyse qualitative data.

5.4 Experiment Four

The work in this thesis was primarily about the game generator tool and not the games. Nonetheless, in this experiment, additional feedback is collected from students to evaluate the suitability and the educational potential of the generated games. This is inline with the early works by Khenisi [117], Eaachak et al. [72], and Marchiori et al. [150]. In this experiment, the games generated using RGG were evaluated by CS1 students to evaluate their educational potential. Students played the given games and answered the Game Experience Questionnaire GEQ. In addition, they completed open ended questions to give final comments about the played games.

5.4.1 Purpose

The purpose of this online experiment was to evaluate the generated custom games with CS1 students. We wanted to elicit feedback from the learners'perspective [249] on the suitability and educational potential of the generated games to enable learning of the difficult topic of recursion. This experiment had three specific objectives, namely: (i) to assess the user experience of CS1 students after playing the games created from the prototype game generator tool; (ii) to find out which game design features CS1 students found most useful; and (iii) to investigate CS1 students'overall opinions about the educational potential of the generated games to enable students to learn the difficult topic of recursion. A CS game is considered educational if the game *"activity teaches a computer science topic, that is, the student understands a topic better or is more skilled at a task after playing the game"* [82, p. 2]. It is worth noting that the main contribution of this thesis is the prototype game generator tool called RGG and not the games. Nonetheless, similar to the prior works by Gaeta et al. [76] and Perez-Colado et al. [210], we conducted an additional experiment with students (secondary users) to demonstrate the suitability and educational potential of the generated games.

5.4.2 Participants

The participants of this experiment were CS1 students from the Faculty of Science in the Department of Computer Science at the University of Cape Town in South Africa. The target audience was chosen because the students had just completed the CS1 course and definitely had encountered the difficulty of learning the recursion topic.

5.4.3 Materials

In this experiment, the following materials were used:

(i) the Game Experience Questionnaire;

(ii) an online informed consent form;

(iii) an online task sheet; and

(iv) a computer or laptop with a web browser and internet connection.

5.4.4 Tasks

The following tasks were performed by participants:

(i) download the games shared through Google drive link;

(ii) play the downloaded game;

(iii) complete the Game Experience Questionnaire; and

(iv) answer the open ended questions on the overall opinion about the potential of the generated games to enable students to learn the recursion topic.

5.4.5 Procedure

This study was conducted online given the Covid 19 pandemic in South Africa at the time. The researcher and three volunteer CS1 lecturers from the University of Cape Town independently generated four game instances each using the prototype game generator tool. Two games were created from the DnD game example and the other two from the Mushroom Picker example. A total of 16 games were created, eight were instances of the DnD game and the rest those of the Mushroom Picker game. Afterwards, the researcher selected half of the generated games. Four games were picked from the generated DnD games and another four from the Mushroom Picker games. The criteria used to select the eight games ware: (i) the game represents an attractive UI, (ii) the game is downloadable, (iii) the game is playable, and (iv) the game can potentially demonstrate the concept of recursion. The eight selected games were assigned to the 20 participants. Every participant received a pair of games through a Google drive link. Each pair had one DnD game and one Mushroom Picker game instance.

(i) The researcher sought ethical approval to conduct the experiment with CS1 students from the University of Cape Town;

(ii) through the CS1 course convener, the researcher sent invitations to participants;

(iii) willing participants signed up for the study;

(iv) before the experiment, willing participants signed an online informed consent form;

(v) a Google drive link to the folder containing a pair of the generated games was shared with participants in each group;

(vi) participants downloaded the shared games;

(vii) participants played the downloaded games offline;

(viii) participants rated the played programming games using the Game Experience Questionnaire;

(ix) participants completed the question on the game design features they found most useful;

(x) participants completed open ended questions on final or general opinions about the potential of generated games to learn recursion;

(xi) participants submitted the completed survey; and

(xii) participants were rewarded 50 Rands for their time.

5.4.6 Criteria Used to Respond to the Third Research Question

RQ 3. *What is the suitability and educational potential of the generated games to enable students to learn the topic of recursion?*
In order to answer the third research question, students played the generated games and reported on their experiences and the games'educational potential. In this regard, three metrics were measured, namely the: (i) user experience of CS1 students after playing the games created from the prototype game generator tool; (ii) game design features CS1 students found most useful; and (iii) CS1 students'overall opinions about the educational potential of the generated games to enable them to learn the difficult topic of recursion. Three sub-questions were posed as follows:

1. *What is the experience of CS1 students after playing games created from a prototype game generator tool called RGG to learn recursion*?

2. *Which game design features did CS1 students find most useful*?

3. *What is the overall opinion of CS1 students about the potential of the generated games to support students to learn the topic of recursion*.

To evaluate the educational potential of the generated games, the Game Experience Questionnaire method was used. The GEQ has three modules, namely (i) the core questionnaire, (ii) the social presence questionnaire, and (iii) the post-game module [103]. Students'game play experiences were assessed using the core GEQ questionnaire based on seven components:

(i) ***Immersion*** - a measure of the degree of a player's involvement or engagement with a game [42, 92] hence direct positive effect on perceived learning [92] ;

(ii) ***Flow*** - a measure of a player's concentration, challenge and skills leading to enjoyment and perceived learning [92, 251];

(iii) ***Competence*** - a measure of the skills acquired by the learner as a result of playing an educational game [92];

(iv) ***Positive affect*** - a measure of the absence of in-game experiences like frustration, disappointment, irritation, and anger or post-game experiences such as regret, guilt, disappointment, anger and revenge [203];

(v) ***Negative affect*** - a measure of the presence of in-game experiences like frustration, disappointment, irritation, and anger or post-game experiences such as regret, guilt, disappointment, anger and revenge [203];

(vi) ***Tension*** - a measure of the extent to which players are frustrated with a game [83];

(vii) ***Challenge*** - a measure of the extent to which players feel that they can progressively accomplish game learning tasks. It means striking a balance not designing too difficult game learning activities to enable students to build on what they already know and at the same time taking care not to introduce too easy activities that could potentially bore the learners [270, 82].

Hamari et al., found that challenge has a direct positive influence on immersion, engagement and perceived learning [92]. In addition, they found a positive association between engagement and perceived learning. Gibson and Bell suggested that the extent to which a game is engaging is one of the criteria used for evaluating games for teaching CS [82]. In this experiment, we wanted to understand whether or not the played generated games and the designed learning tasks challenged participants enough to enable them to learn programming [92] within their Zone of Proximal Development [270]. It is believed that such user experiences could have a great bearing on the adaptation potential of the GBL approach to induce learning programming in CSE.

5.4.7 Data collection and analysis

The Game Experience Questionnaire was used to collect data on the subjective experience of CS1 students after playing the games generated from the prototype.

- Likert scales were used to collect overall subjective user experience using the Game Experience Questionnaire. Scoring guidelines for the core GEQ module

were used as suggested by Ijsselsteijn et al., [103]. Each of the seven component or measure scores were calculated as the average values of its items [183, 103, 184].

- Qualitative comments were coded and analysed thematically.
- SPSS was used to analyse quantitative data.

5.5 Summary

This Chapter presented the protocol used for the empirical experiments conducted to evaluate the prototype game generator tool (the Recursive Game Generator - RGG) and the generated games. A description of the aim, objective, and participants for each experiment was provided. In addition, the Chapter summarised the tasks performed by participants, materials used, procedure used and the study design. The Chapter also discussed the evaluation criteria used to answer the second and third research questions. Finally, data collection and analysis methods was summarised. Next, Chapter 6 presents and discusses findings from the evaluation experiments.

CHAPTER 6

Results and Discussion

6.1 Introduction

This Chapter presents and discusses the results obtained from four empirical evaluation studies conducted in this thesis. Three are with a prototype game generator tool to support CS1 teachers in higher education and one with the generated games to help students learn the topic of recursion. The results of experiments one, two and three provide empirical evidence that answer the second research question: "*How effective is the use of a generator tool by CS educators in creating games to teach recursion*"? Meanwhile, results of experiment four provide evidence that answer the third research question: "*What is the suitability and educational potential of the generated games to enable students to learn the topic of recursion*"? Findings of experiment one are presented in Section 6.2. This is followed by those of experiment two in Section 6.3. Section 6.4 is a summary of results of the experiment three. Findings of experiment 4 are summarised in Section 6.5. Lastly, comparative analysis of participant cohorts results is in Section 6.6.

6.2 Experiment One

This experiment was conducted with CS1 instructors to provide empirical evidence of the effectiveness of the prototype game generator tool. This was done by evaluating it's usability. Metrics such as ease of use, usefulness, learnability, and user satisfaction were measured. In addition, post-task ratings including task difficulty, task success, and the estimated time on task were measured.

6.2.1 Demographics

30 experienced CS1 instructors from Kenya and South Africa took part in this study. A total of 60% of the respondents considered themselves as gamers while only 20% had used games in teaching. Table 6.1 is a summary of other demographic information.

Table 6.1 Demographic information of Experiment one

Variable	Category	Freq	Percentage %
Age	25 - 30 years	2	6
	31 -35 years	5	16
	36 -40 years	8	27
	41 -45 years	11	38
	46 -50 years	3	10
	Above 50 years	1	3
Education	Bachelor degree	2	6
	Masters degree	20	67
	PhD. or higher	8	27
Institution	High school	2	6
	Tertiary college	4	14
	University	24	80
Experience	Less than 5 years	2	6
	5 -10 years	12	40
	11 -15 years	11	38
	16 -20 years	3	10
	Over 20 years	2	6

6.2.2 Task difficulty

For all the post task rating Tables and Figures, the tasks are: Task 1 - creating an account and logging in; Task 2 - creating a custom game using a given example; Task 3 - customising the created game, and Task 4 - downloading the created game. Figure 6.1 shows a visualisation of the results in a stacked bar chart. Overall, the findings suggest that most users found the tasks easy to perform with the RGG tool, with creating user account/ logging in (94%) and downloading the created games (91%) being ranked as the easiest. This was followed by creating a custom game using a given example (74%) and lastly customising the game (71%). 20% found creating a game using an example difficult.

6.2.3 Task success

Table 6.2 presents the task success results, where 'failure 2'stands for 'failure-thought the task was easy but it was not', 'success 1'stands for partial success with assistance, 'success 2'stands for partial success without assistance, 'success 3'stands for complete success with assistance and 'success 4'stands for complete success without assistance. The values in the table are the number of participants who considered themselves suc-

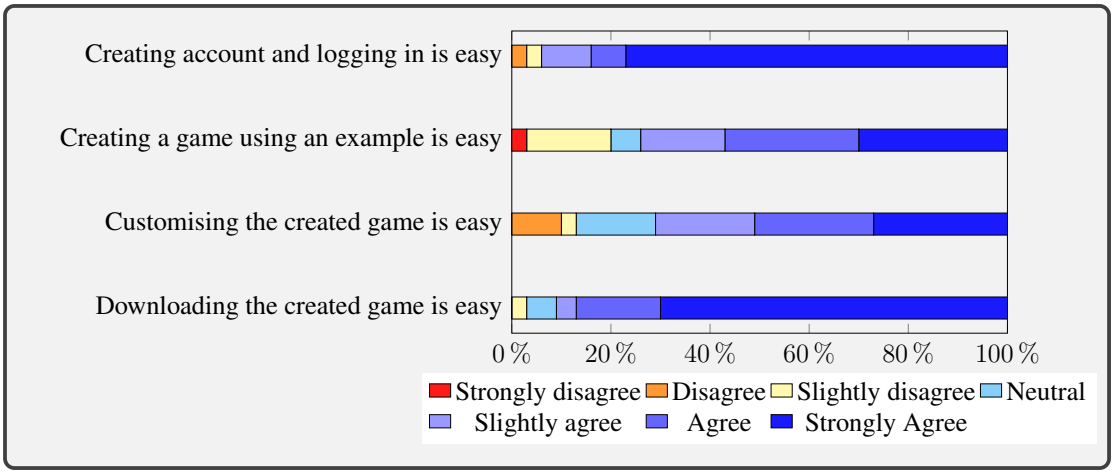


Fig. 6.1 Task Difficulty Rating

cessful on the tasks in each task success level. The findings suggest that participants were successful in most tasks. Considering the sum of participants who successfully completed the tasks with or without assistance, 27 (90%) successfully created a user account and logged in. This was followed by downloading the created game 25 (83%), creating a game using an example 23 (77%), and lastly customising the created game 20 (66%). Meanwhile, eight participants (27%) were partially successful in task 3. This was followed by five participants (17%) on tasks 2 and 4 respectively. In addition, three participants (10%) reported partial success on task 1. Two participants (6%) reported failure in tasks 2 and task 3. Finally, no failure was reported in task 1 and task 4.

Table 6.2 Number of participants successful on the tasks under various task success levels

	Failure gave up	**Failure 2**	**Success 1**	**Success 2**	**Success 3**	**Success 4**
Task 1	0	0	2	1	24	3
Task 2	0	2	2	3	11	12
Task 3	1	1	5	3	10	10
Task 4	0	0	2	3	22	3

6.2.4 Estimated time on tasks

We asked participants to give an estimate of the time they spent while performing each of the four tasks to measure the tool's efficiency. This was done on a 5 point scale. Time ranges (discrete time intervals) were used, where 1 represented 10 to 13 minutes. 2 represented 7 to 10 minutes, 3 represented 5 to 7 minutes, 4 represented 3 to 5 minutes

and 5 represented 0 to 3 minutes. Figure 6.2 is a visualisation of the spread of completion times by all users presented as frequencies of participants in each scale. Results indicate that 23 (77%) users spent 0-3 minutes in task 1. In task 4, 15 (50%) users spent 0-3 minutes. Meanwhile, 5 (17%) users estimated spending 10 - 13 minutes in task 2. In task 3, 4 (13%) teachers said that they spent between 10 - 13 minutes. No participant spent more than 7 minutes in task 4.

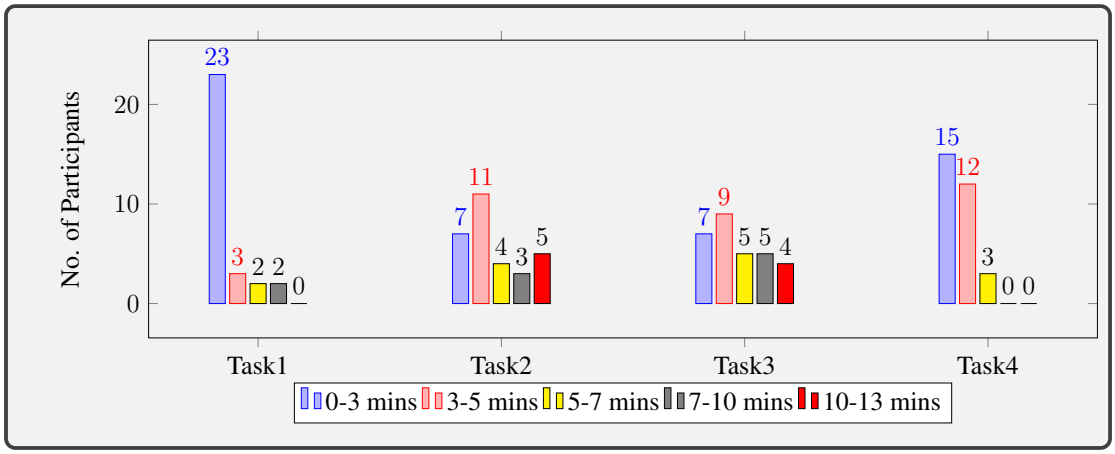


Fig. 6.2 Estimated Time on Task Rating

6.2.5 Usefulness

To evaluate the usability of the RGG tool, respondents answered the USE questionnaire [135], which has four dimensions: Usefulness, Ease of use, Ease of learning and Satisfaction. Participants'ratings of the tool's usability on these dimensions were scored from 1 for strongly disagree to 7 for strongly agree. For the usefulness dimension, results obtained suggest that most CS1 lecturers surveyed answered affirming the usefulness of the RGG tool. As can be seen in the stacked bar chart in Figure 6.3, overall, 87% agreed that the tool would be useful for helping programming instructors to create games to teach the recursion topic . Another 94% of the participants agreed that the tool would make the things programming instructors want to accomplish with the tool easier to get done. 84% said that the tool would save them time when creating games to teach programming. Lastly, 81% reported that the RGG tool would either make instructors more productive in their teaching, more effective or give them control over their teaching activities. However, some 10% of the users found the tool not useful in saving time, suggesting possible improvements in future. These results demonstrate the pedagogical promise of the RGG game generator tool in teaching programming in higher education.

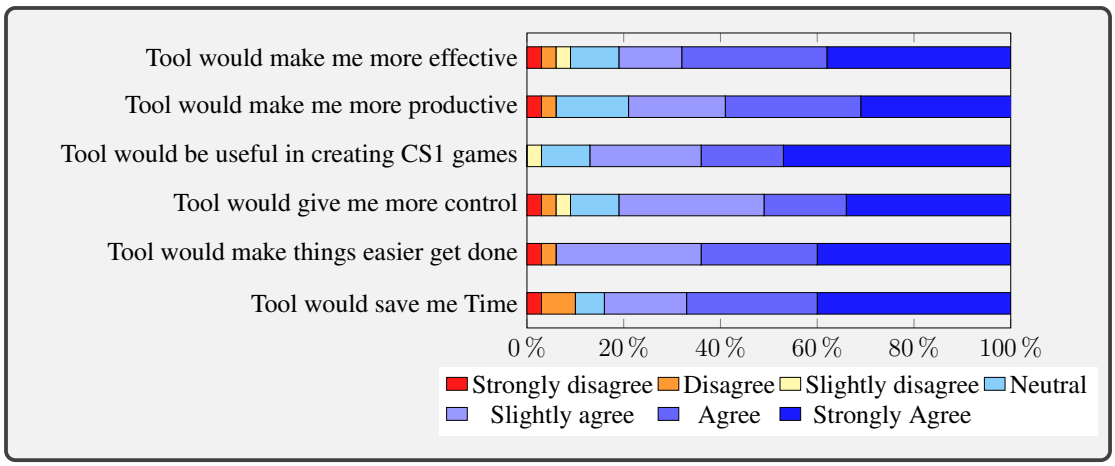


Fig. 6.3 Perceived Tool Usefulness

6.2.6 Ease of Use

Figure 6.4 presents the subjective rating of the tool's ease of use by participants. Most participants found the tool easy to use (87%), simple to use (81%), usable (80%) and requiring the fewest steps to accomplish tasks (74%).

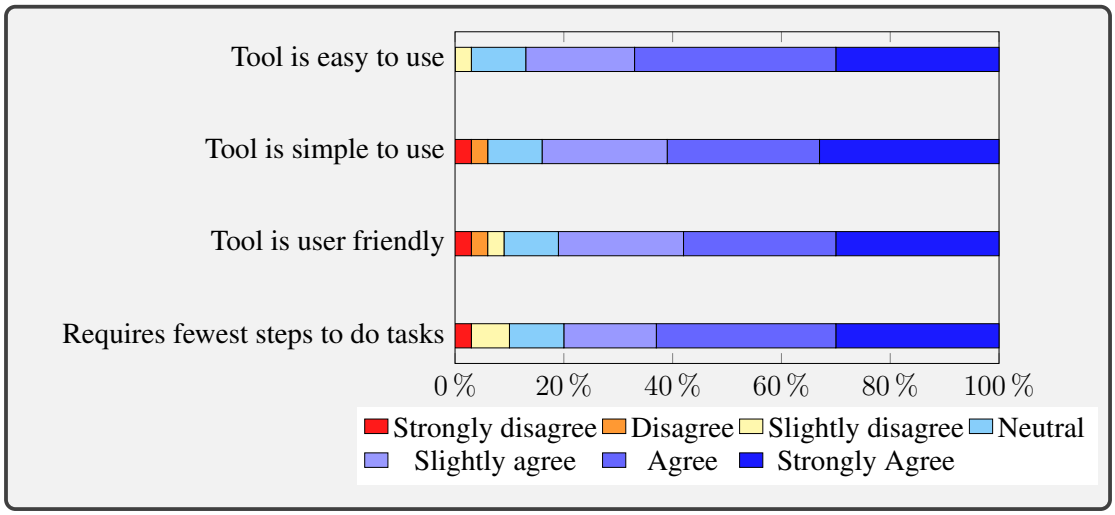


Fig. 6.4 Perceived Ease of Use

6.2.7 Ease of Learning

When asked to evaluate the usability of the tool on the learnability dimension, results suggest that, overall, participants found RGG easy to learn. In total, about 80% of the respondents said that they either learnt the tool quickly or that they could easily remember how to use it. Another 77% reported that it was easy to learn how to use

it. Generally, these results suggest learnability of the tool. Among the participants who disagreed, 13% reported that it was neither quick nor easy to learn to use the tool. Figure 6.5 presents the subjective rating of the tool's ease of learning by the participants.

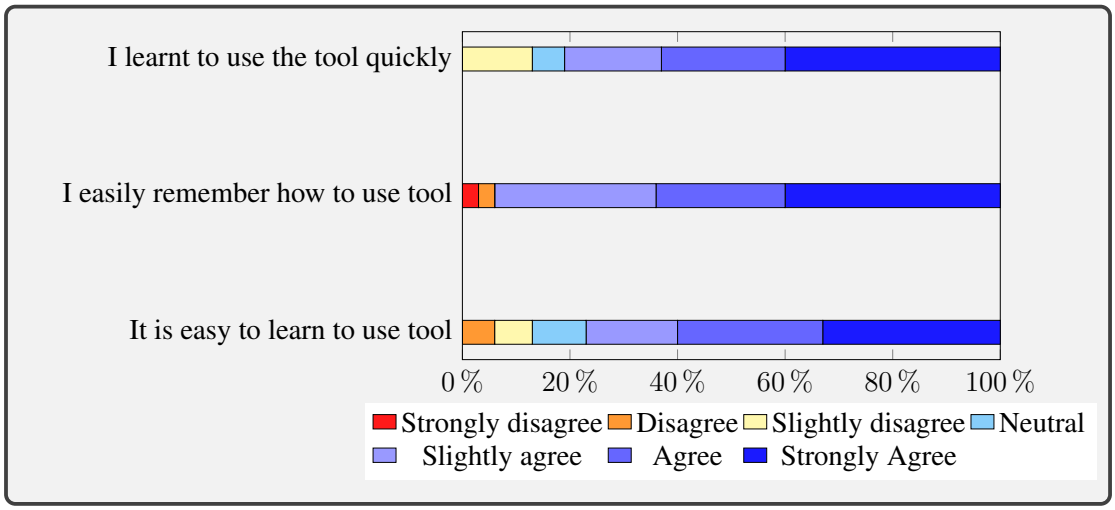


Fig. 6.5 Perceived Tool Ease of Learning

6.2.8 User Satisfaction

We asked the participants the extent to which they agreed or disagreed that they were satisfied with the tool based on the satisfaction items in the USE questionnaire. Figure 6.6 presents a summary of responses as a percentage. In total, results obtained show that 83% of programming instructors agree that they would recommend the RGG generator tool to a friend with 56% strongly agreeing. Another 81% agreed that they both needed to have the tool at their work place and that it was wonderful. 77% felt that the tool was fun to use. Lastly, approximately 75% of the respondents were either satisfied with RGG or thought it worked the way they wanted it to.

6.2.9 Final user comments

Lastly, we asked respondents to give one final comment about the RGG tool to gain more insight about their impressions. Respondents were of the opinion that RGG was a useful and noble tool for supporting teaching programming in higher education (80%). In addition, a majority (75%) strongly recommended its adoption in teaching programming in higher education. Below are some direct exemplars:

- *"It's a useful learning tool to us lecturers and students."*

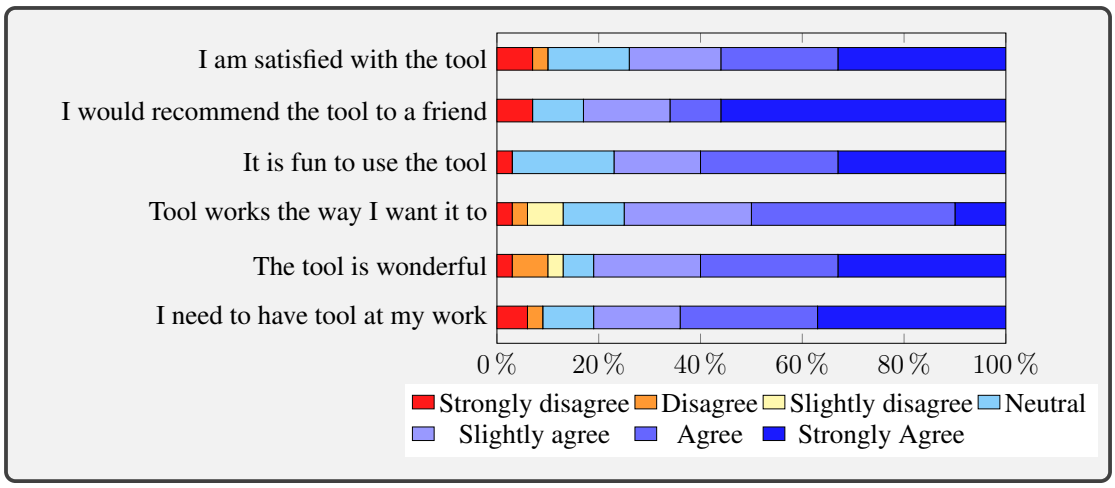


Fig. 6.6 Perceived User Satisfaction

- "*This is a great tool that can help students to connect class concepts with real applicability.I would recommend it to be improved further and adopted for teaching.*"
- "*I think it is lovely and a lot of fun - I think it would help a lot of students and lecturers and maybe entice some of the lazier kids to get involved.*"
- "*I strongly recommend it for adoption in teaching Recursion in Computer Programming.*"
- "*It's a brilliant tool for teaching the concept of functions and recursion in particular.*"
- "*The research is very useful and it will enable our students to learn how to use such algorithms by implementing them in games applications.*"

It is worth noting that while most of the teachers had good UX with the RGG tool and gave positive final comments, there were also some negative comments. Among others, some teachers were not happy with: (i) the user interface, (ii) the log-in and authentication process, (iii) game feedback mechanism, (iv) the auto-marker feature, (v) gender bias in design, and (vi) use of only python language. While the comprehensive details of the negative comments are in Appendix C10, bellow are some exemplars:

- *"The interface is not intuitive."*
- "*The authentication process is too strict for a teaching tool.*"
- "*Does not provide hints on how to solve problems.*"
- "*Lack of ability to customise more parts of it, for example the editor, language..cannot even choose Python 3 which is more recent.*"

- "*I think it might be a bit of a miss for girls..we often forget about females when we thin programming*."
- "*Help Section is confusing and it takes effort to notice and remember relevant parts*."
- "*I could not figure out how to create my own games*."
- "*It does not allow full programming by students*."
- "*Instructions to students in the given examples is really fuzzy*."

One particular teacher who was extremely negative was not convinced that the games generated by the prototype could teach the recursion concept to students. Following is an excerpt of the comment from the teacher as demonstrated in Appendix C11.

" *Overall, I don't think that the games that this tool generates would teach students recursion more effectively than the usual assignments given in Computer Science courses. To be blunt, I think this is just a nice way to "gamify" any programming task, not just recursion. There are actually games out there on the market, that I have played, that teach much more about the concepts behind recursion than this tool, unfortunately....I feel that the creator of this tool must definitely do some more research on current games on the market that can teach programming (since these would also be available to universities, for a price), as well as academic literature perhaps (although I am not well-versed in this area of Computer Science at all) and make design decisions for the Recursive Game Generator Tool informed by what is currently state of the art*".

6.2.10 Discussion

The teacher who said that the generated games cannot be compared with professionally designed games in the market was actually right. There is no way this comparison can be done. This is because the games in the market are designed by professional game developers but the ones in this thesis are generated. Also, the software is merely a prototype designed to test the idea of a game generator. Furthermore, as the teacher rightfully claims, commercial games are readily available. However, they may lack pedagogy in their design and are generally not easy to customise, but teachers always like tailoring their learning material to the needs of their teaching contexts. The aim of this thesis was to aid non-technical teachers with a game authoring tool that could enable them to easily create educational games.

Partial success means that the users did not completely succeed in performing the tasks neither did they fail [8]. This provides additional useful information that cannot be obtained when we use only binary success. Such additional information may include:

(i) why some users failed a particular task and (ii) which particular tasks users needed help [8]. In the context of this experiment, it seems that users required additional help in task 3 followed by tasks 2 and 4 respectively. Meanwhile, users required least help in task 1.

Overall, findings from this study's usability evaluation with educators agree with those of previous similar work [262]. The finding that the prototype can generate games that can allow students to practice coding is in line with the design principle for programming games [48]. For games generated from the Mushroom picker example, upon completing the code snippet, students can visualise the execution of their code/ algorithm through a character moving recursively as recommended by [48] and Eagle and Barnes [73]. Additionally, the games created from the Mushroom picker example use simple commands (LEFT(), RIGHT(), UP(), DOWN()) for novices in sync with the ALICE programming game [56]. The game generator prototype not only extends these design principles; it also authors customisable games in line with recommendations by Marchiori et al. [150].

One limitation of this experiment was the missing feedback from students. However, the primary users of the proposed tool are CS1 educators who interact directly with it. Students are merely secondary users who are affected by the capability of the primary users to carry out tasks with the tool [143]. Nonetheless, experiment four addressed this concern. Consequently, by testing the idea of a game generator tool using a prototype and evaluating it with instructors, the positive findings on usability and general comments potentially demonstrate that the tool could be suitable for adoption by CS1 instructors to support teaching novices. Consequently, this answers research question two.

6.3 Experiment Two

This experiment sought to provide additional empirical evidence of the effectiveness of the prototype game generator tool to support CS1 educators as well as highlight key design features. It had two contributions:

1. *It provided additional empirical evidence of the perceived usefulness of the prototype as a support tool for CS1 teachers, further demonstrating the adoption potential*; and

2. *It identified key game authoring features that could lead to design strategies for future serious games development.*

Table 6.3 Demographic information of Experiment two

Variable	**Category**	**Freq**	**Percentage** %
Courses taught	Only CS1	16	57
	CS1 and advanced courses	12	43
Game Programming skills	None at all	7	25
	Low	15	54
	High	6	22
Experience	Less than 5 years	7	25
	5 -10 years	9	32
	11 -15 years	6	21
	16 -20 years	3	11
	Over 20 years	3	11
Used GBL before	Yes	9	32
	No	19	68

6.3.1 Demographics

The number of participants in this experiment were 28. The distribution by country was such that 18 came from Kenya, two - South Africa, three - Malawi, two - United States of America, and one from Brazil, Finland and Ireland respectively. Furthermore, the subjects varied by (i) CS programming courses they teach, (ii) game programming skills, (iii) teaching experience in years, and (iv) whether or not they had taught using games. Table 6.3 is a summary of the demographic information.

6.3.2 Usefulness and suitability of the prototype

In the first experiment [16], we reported on CS1 educators'perceived usefulness of the prototype using the Usefulness measure defined by Lund in the USE questionnaire [20]. In the second experiment, additional usefulness items were developed to get a broader sense of the prototype usefulness according to CS1 instructors (the potential adopters). Figure 6.7 is a visualisation of the responses.

The sum of percentage responses of the agree and strongly agree categories suggests that participants believe that the generated games are easy to distribute (75%). This was followed by the revelation that the prototype increases integration of the educational value and that the generated games could support student centered learning (72%). In the third place was the feeling that it reduces game production cost and effort and that the generated games support student centered learning (71%). It is also positive to note that about 68% of the participants thought that the prototype is useful to both students and teachers and that the generated games are useful for teaching the recursion topic in

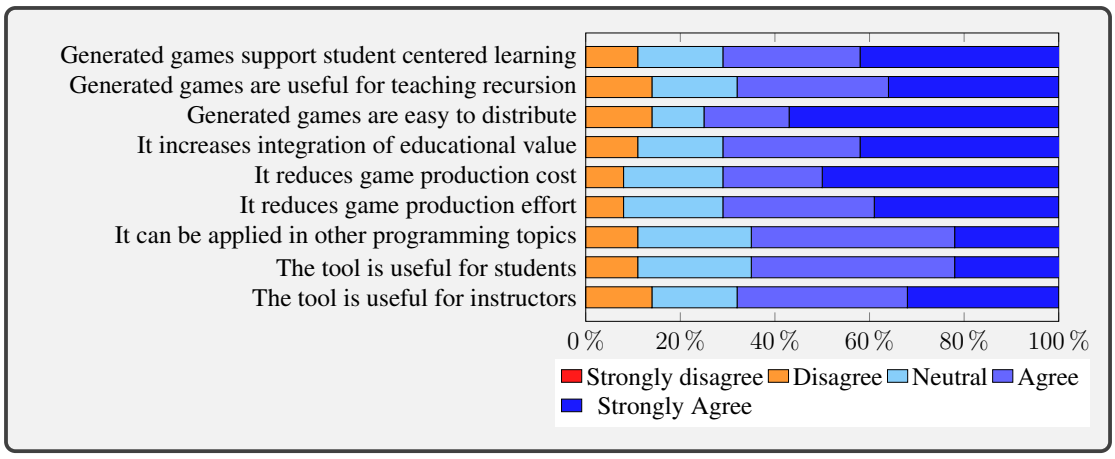


Fig. 6.7 Responses to Additional Usefulness items

CS1. A similar percentage equally believed that the prototype could be applied in many other CS1 topics. Overall, the results were in line with findings of the first experiment [16].

However, instructor responses on all items revealed one surprising finding that a significant number (18%) were neutral. We expected that respondents would either agree or disagree. One possible reason could have been the fact that most instructors have not used game authoring tools before. Or maybe, they are not strong programmers themselves. Therefore, they could not easily make judgments.

6.3.3 Prototype's most useful features

To understand instructors'views about the prototypes's most useful features, respondents chose from a list of 13 features. Participants were allowed to select more than one. Table 6.4 presents the number of participants and the percentage for each feature. Going by the percentage of participants, results show that the top five most useful features were:

(i) Custom game generation (64%)

(ii) Built-in game generation template (57%)

(iii) Support for common operating systems (54%)

(iv) No game programming required (50%)

(v) Game download and distribution support; Integrated Development Environment (IDE) in the generated games; and Character customisation (46%)

Table 6.4 Ranking of the prototype's perceived useful features

Feature	Participants↓	Percentage↓
Custom game generation	18	64%
Built-in game generation template	16	57%
Supports common operating systems	15	54%
No game programming required	14	50%
Game download and distribution support	13	46%
IDE in the generated games	13	46%
Character customisation	13	46%
Player reward and motivation	12	43%
Built-in assets	11	39%
Built on top of Unity engine	11	39%
Help feature	11	39%
Database file storage	9	32%
Works with devices that support coding	7	25%

6.3.4 Final comments

Participants were given an opportunity to express their final general comments about the prototype and the proposed game generation tool. This feedback was captured using the open ended question: "*What is your general comment about the game generator tool*"? Fifty (50) responses were coded into eight categories or themes. Table 6.5 shows the results. Overall, instructors indicated that the game generator tool was a good and an innovative idea 15 times (30%). Respondents also commented about the usability of the prototype, noting that it was useful 15 times (20%) and easy to use and saves time (seven times, 14%). Additionally, teachers reported that the generated games were interesting and engaging eight times (16%). They also said that they would recommend the tool to colleagues seven times (14%). One response (2%) noted the fact that the design could support common operating systems. The Mushroom Picker game's design was lauded as great given its ability to also teach loops (2%). However, one response noted that students may have problems with the last level regarding recursion. While the rest of the raw data is found in Appendix C12, below are examples of two quotes:

"*It is a promising tool and the future customization of games and their problems will be fantastic. However, I would like to call attention to some of its characteristics. 1) The Mushroom picker is great. It's able to be used to help students to understand LOOP also. In the case of the last question (regarding recursion), as the students cannot see all the paths and their obstacles, they may have some problems reaching a good solution. 2) In my opinion, students may experience a cognitive load on trying to understand what they have to do to solve the Runner game".*

"*It is a timely and very useful game generator tool that makes it easier for me to create customised games to assist in teaching the topic of recursion. Its Help, built-in game generation template, Custom game generation, game download and distribution support features stand out in making it very easy to learn and use. I also like the fact that it has support for common operating systems (Windows and Linux). I highly recommend it for widespread distribution and use*".

Table 6.5 Top 8 themes from final general comments

#	Theme	Frequency↓	Percentage↓
1	Good and innovative idea	15	30%
2	Useful	10	20%
3	Generated games interesting and engaging	8	16%
4	I recommend the tool	7	14%
5	Easy to use and saves time	7	14%
6	Cognitive load with Runner Up game	1	2%
7	Support for common Operating Systems	1	2%
8	Students can't see all parts in Mushroom-Picker Last Question	1	2%
	Total	50	100%

6.3.5 Discussion

The problems identified in the last level of the Mushroom Picker game and Runner games provide opportunities for improvements. For instance, the Runner game was implemented using procedural content generation but was not fully designed due to time constrains. Nevertheless, there is the possibility to complete the game and fully implement procedural content generation in future work [166]. Moreover, participants were clearly instructed in the task sheet to only evaluate the Mushroom picker and the DnD games. However, it seems that some participants were eager to test all the three games.

Overall, this study's results suggest that the instructors had a positive user experience with the prototype game authoring tool. The fact that instructors found it useful that the generated games are easy to distribute addresses the deployment stage adoption issues (game delivery and distribution) raised by Torrente et al. [261]. Likewise, it responds to the ease of installation and running evaluation criteria of CS educational games suggested by Gibson and Bell [82]. The finding that the tool increases integration of the educational value and that the generated games can support student centered learning attempt to tackle the design stage issues of harmonising educational value with fun [228, 261]. Furthermore, results show that the instructors agreed that the prototype is useful with respect to reduction of game production effort and cost. This helps in

Table 6.6 Demographic information of Experiment three

Variable	**Category**	**Freq**	**Percentage** %
Age	18 - 20 years	14	63
	20 -22 years	6	27
	22 -24 years	1	5
	Above 24 years	1	5
Teaching subject	Maths/ Computer	19	85
	Chem/ Computer	1	5
	French/ Computer	2	10
Game prog. skills	None at all	2	10
	Low	10	45
	High	10	45
Gamer	Yes	17	77
	No	5	23

resolving the reported production stage barriers (cost, time, technicality) in previous works [4, 261].

Finally, the finding on the prototype's most useful features (custom game generation, built-in generation and that no game programming is required) suggest that instructors with basic computer skills could easily create and customise their own educational games. This could enhance the adoption of GBL in mainstream teaching. We argue that the findings of experiment two provide additional empirical evidence demonstrating the effectiveness of the prototype game generator tool, hence answers research question 2.

6.4 Experiment Three

To further demonstrate the effectiveness of the proposed tool, the third experiment attempted to investigate the subjective usability and user experiences of CS trainee teachers when using the prototype game generator tool called RGG to create games to teach the recursion topic. The standard Attrakdiff questionnaire [94] was used.

6.4.1 Demographics

The sample of the main study varied by age, teaching subjects, interest in games, and learning with game experience, among other things. A total of 63% of the final study participants fell within the 18 to 20 age bracket. A total of 85% had registered for mathematics and computing as their two teaching subjects. A total of 77% considered themselves as gamers. A total of 95% had not been taught any course using games before while 60% did not study computing in high school. Results of the demographic information are summarised in Table 6.6.

Table 6.7 Mean values

Dimension	DnD game	Mushroom picker game
PRA	0.95	1.64
HQ-I	1.36	1.68
HQ-S	1.32	1.68
ATT	1.73	2.01

6.4.2 Scale mean scores

The mean values for the four AttrakDiff dimensions (scales) were arrived at by averaging the values of all answers inside each dimension. Table 6.7 presents the details and the graph in Figure 6.8 is a visualisation. In the graph, the vertical axis shows the average assessment values of word pairs inside each group while the horizontal depicts the four word groups/ dimensions. The dimensions are PRA, HQ-I, HQ-S, and ATT.

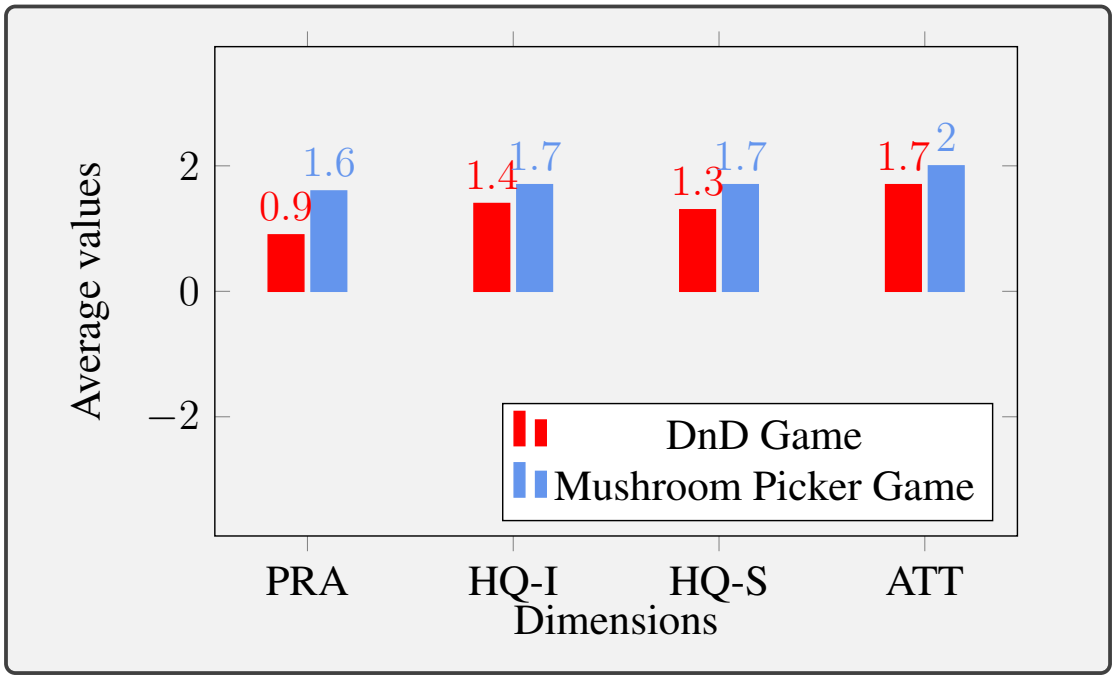


Fig. 6.8 Attrakdiff dimensions average values

Overall, the UX reported by participants suggests a good experience when creating custom games from both the DnD and Mushroom Picker game examples. The graph shows that the values were above zero for all dimensions. However, the attractiveness dimension was rated the highest by all participants for the prototype game generator. This suggests that respondents found the prototype's design attractive and likeable. Moreover, participants appeared to have had a positive general impression of the prototype. The pragmatic quality results suggest that users found it easy to understand how

to use the prototype and felt in control of their interactions with it, especially, when creating custom games from the dynamic Mushroom Picker game example.

The HQ-I and HQ-S mean scores were similar for the Mushroom Picker game example and almost the same for the DnD game. This reveals that users found the prototype equally stimulating in terms of novelty, content and interaction when using it to create custom games from both game examples. Additionally, it suggests that playing the generated games was straight - forward, brought players closer to the game play and connected them to other players, particularly, for the dynamic Mushroom Picker game example. Results for the Hedonic quality are promising, given the prominence the CS research community has accorded it [64].

6.4.3 Pragmatic Quality

Figure 6.9 presents the mean values of word pairs of PRA and HQ-I dimensions of the Attrakdiff questionnaire. PRA represents word pairs "technical - human" , "complicated - simple", "impractical - practical", "cumbersome - straight forward", "unpredictable - predictable", "confusing - structured", and "unruly - manageable". It measures the usability of a product in terms of how successfully a user can use a product to achieve her/ his goals [154]. Results suggest that using RGG to create custom games from both DnD and the Mushroom Picker game examples gave trainee teachers a positive usability experience. However, the experience was more positive for the Mushroom Picker game example than the DnD game. As can be seen from Figure 6.9, the prototype was clearly structured and manageable. This suggests that users were in control of their interaction with it and found the user interface organised.

6.4.4 Hedonic Quality Identity (HQ-I)

Regarding HQ-I dimension (Figure 6.9), respondents rated all the items positively. This indicates that participants socially identified [154] with the prototype. This finding is particularly useful given that most participants were within the 18 - 20 and 20 - 22 years age brackets. Additionally, they found the prototype connective, professional, stylish and premium. The high perception of HQ-I towards the use of RGG highlights an interesting result - that is young and inexperienced CS teachers found the prototype socially engaging. This could potentially increase acceptance of the game generation idea [192] among this group of users who may need it most given their limited teaching experience.

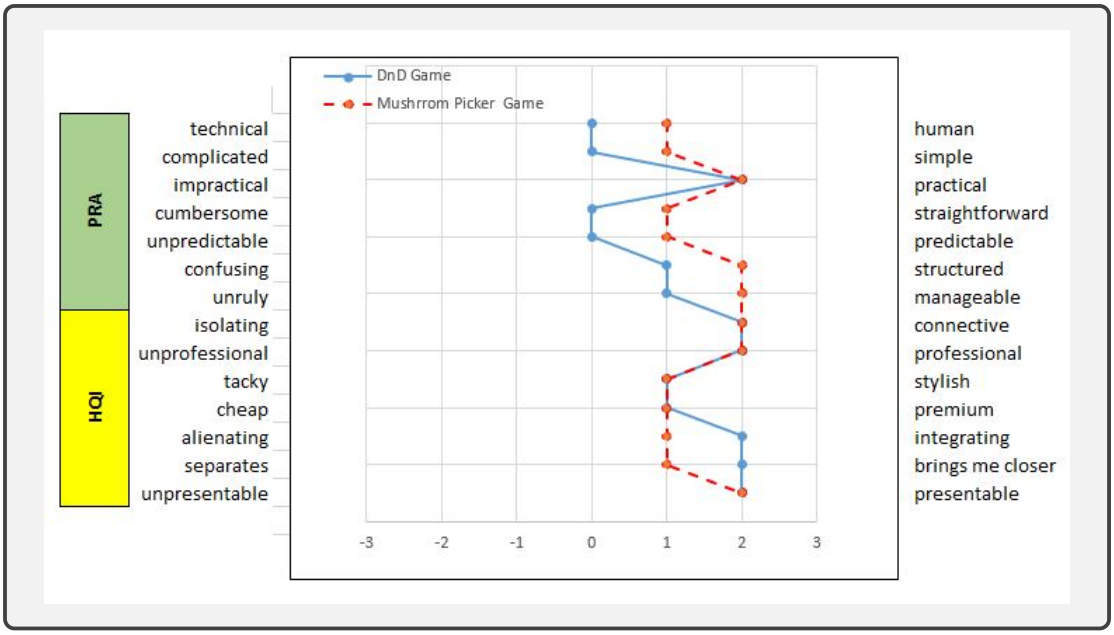


Fig. 6.9 Pragmatic and Hedonic Quality Identity Word Pairs Mean Values

6.4.5 Hedonic Quality Stimulation (HQ-S)

Hedonic Quality Stimulation (HQ-S) indicates the extent to which a product can support user needs relating to novelty, content, stimulation, and presentation of style [154]. Figure 6.10 shows that users found the prototype inventive, creative, bold, innovative, captivating and challenging. This result suggests that the prototype was perceived by users as novel and stimulating.

6.4.6 Attractiveness

Concerning the attractiveness dimension, respondents found the prototype attractive, likeable, appealing and good when creating and playing custom games from both the DnD and Mushroom Picker game examples, as depicted in Figure 6.10.

6.4.7 Qualitative comments

To gather more insights about the prototype, we asked respondents to answer two open ended questions. The first was about the bugs / technical issues they encountered during the usability test and any design suggestions. The second was on their final opinions. Regarding bugs/ technical issues, two themes arose: (i) prototype hosting - Internet speed; and (ii) syntax errors. Fifty five percent of the participants reported slow download of the generated games when testing the prototype. However, this had nothing to

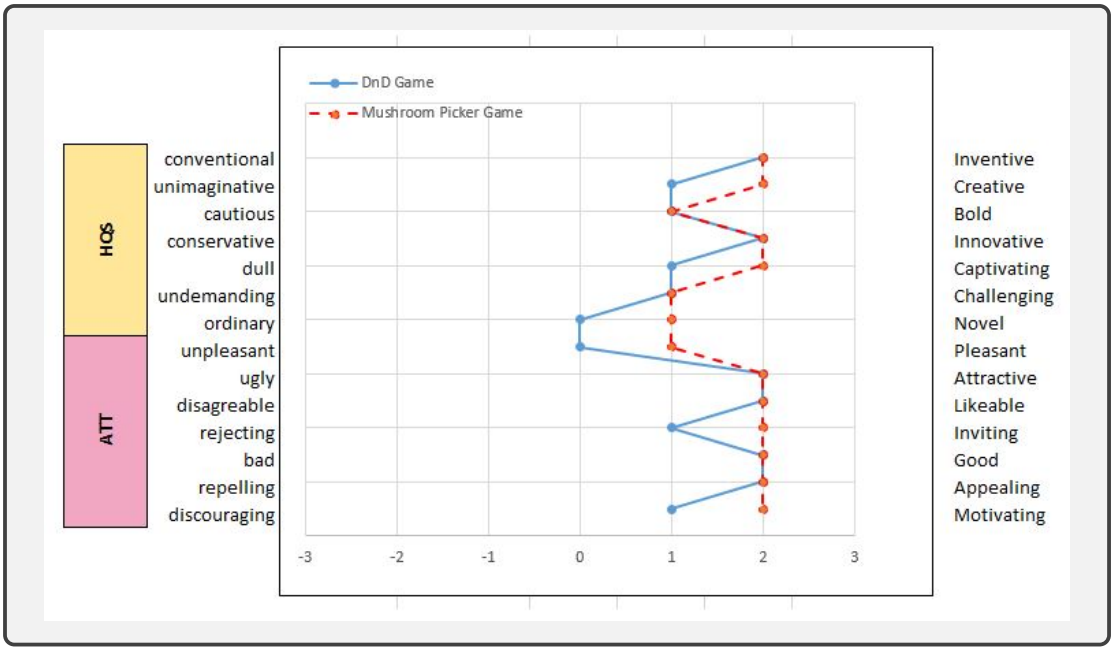


Fig. 6.10 Hedonic Quality Stimulation and Attractiveness Word Pairs Mean Values

do with the prototypes's hosting but rather the slow Internet speed in the laboratory. In addition, some had challenges with Python syntax since they had not encountered programming with the language. One participant suggested the use of different colour code segments to ease code readability and debugging. Findings of the second question indicate that words like: (i) game(s); (ii) generator; (iii) good idea; (iv) teaching; (v) teach; (vi) programming; (vii) help; (viii) learners; (ix) teacher; and (x) students appeared more times (had the highest percentage). Raw data is available in Appendix C12.

Overall, thematic content analysis revealed that 70% of the respondents found a game generator tool a good idea for supporting trainee CS teachers who may wish to adopt game based learning during their teaching practice or when they start teaching. Another 65% noted that the generated games were interactive, practical, interesting and fun for learning programming. Some direct quotes follow:

- "*The generator prototype is a convenient tool for generating games, especially for teachers to use as a teaching aid*".
- "*The game generator tool is a noble idea*".
- "*The creation of the game was good and a learning experience at the the same time....*".
- "*Can be used by teachers in teaching because first it is more fun,.. Challenges learners to be also innovative ...*".

- "*It can help students and teachers in programming*".

While a large percentage of participants reported positive user experience with the tool and the generated games, a further analysis of the qualitative comments revealed that some 35% faced some technical and programming challenges. Some of the challenges these participants faced included slow network speed and syntax errors among others. The following examples of negative comments demonstrate these challenges.

- "*Indentation was some how challenging*".
- "*Network speed was very slow*".
- "*Slow network connections*".
- "*Failure to run a problem due to some errors in my program*".
- "*Knowing the exact manner of inputting the solution was tricky and how to indent to in the right way was hard too*".
- "*The game should be more presentable and realistic..use real and attractive features*".

6.4.8 Discussion

The positive findings based on experiences of trainees from this study are promising and a clear indication of the potential adoption of the proposed game generator tool by this group in teaching programming in higher education. Similar to results from a previous study, the finding that participants were excited with the game generation idea and the created games suggests that trainee teachers are most likely to adopt such teaching tools [189].

Regarding development, two issues emerge from the results that could inform design of future serious games or other programming tools. They include the need to consider: (i) different colours in the code snippets design; and (ii) other programming languages. Different colours is perceived to improve code readability and debugging. On the other hand, the finding that participants had a better UX with the Mushroom Picker game due to the visualisation feature is consistent with the work by [48].

Finally, the fact that all generated games have an IDE could be useful to teachers for checking students'problem solving activities and coding behaviour [140]. One limitation for this experiment could be the limited number of respondents. However, 22 participants (70%) could be considered representative enough given the Covid 19 pandemic. Moreover, the respondents had been taught some programming courses as well as how to prepare computing classes. Consequently, they were deemed suitable for the experiment.

Overall, the evidence gathered from this experiment established that respondents had a good UX with the prototype and found the game generator idea to be good. We therefore argue that supporting CS trainee teachers with such game authoring platforms could generally have the potential of advancing the adoption of GBL in CSE education, particularly among this target audience.

6.5 Experiment Four

In order to ascertain that serious games have positive learning impact on players, there is a need to assess game play experiences of such games [216]. This can be done formally by using exams or informally by testing the ability of a player to perform a game task that maps to a particular learning goal [216]. In this section, findings of a user study conducted to assess game play experiences of 20 CS1 students are presented. In this regard, the section provides convincing empirical evidence about the validity of the generated games for learning programming. This is done by demonstrating the educational potential of the generated programming games from the lens of students. The standard Game Experience Questionnaire (GEQ) is used. In addition, participants answer open ended questions.

6.5.1 Demographics

Twenty CS1 students participated in this study. The sample varied by age, CS majors versus non-majors, competency with the Python programming language, gamers versus non-gamers, those that had been taught with games before versus those that had not and the operating system used in the experiment. A total of nine participants (45%) were aged 18 years whereas six (30%) were 19 years old. A total of three participants (15%) were 20 years old. Finally, three (10%) reported being 28 years old. Half of the participants (50%) were majoring in computer science whereas the other 50% were non-majors. Regarding Python programming language competency, one participant (5%) reported being poor, ten (50%) said that they were average whereas nine (45%) thought that they were good in the language. 12 participants (60%) considered themselves as gamers and eight (40%) as non-gamers. Ten participants (50%) had been taught using games before and the remaining ten (50%) had not. Lastly, regarding the operating system used to carry out the experiment, a majority of participants (95%) used windows. Only one out of 20 used Linux. All participants were recruited through the course convener who sent an invitation link to the class mailing list.

6.5.2 CS1 Students'Game Play Experiences

In this Section we present findings of participants'game play experiences based on: (i) the mean score of each GEQ component as suggested by [103, 183, 184], and (ii) the descriptive statistics of each GEQ component.

6.5.2.1 GEQ component mean scores

The GEQ has seven dimensions or components namely: competence, immersion, flow, tension, challenge, negative effect and positive effect. Regarding participants'game play experiences, each of the seven component mean scores were computed as the average values of its items [103, 183, 184]. Participants evaluated their game play experiences on a five point Likert scale. 'Not at all'was assigned the value zero (0) and 'Extremely'the value four (4).

Mean scores for the Mushroom Picker game were highest on the negative effect, tension, and positive effect components. The mean score for negative effect component was 2.61. This was followed by a mean score of 2.60 for the tension component then 2.23 for positive effect component. Lastly, competence had a mean score of 2.11. A high mean score suggests a positive user experience. Meanwhile, a mean score of 1.94 which is close to 2.0 was found for the immersion scale. This is a good indicator that participants were immersed in the game play. The mean score for the flow items was 1.59 while that of the challenge items was 1.44. These two items recorded the lowest mean scores.

Likewise, for the DnD game, highest mean scores were reported on the negative effect, tension, competence, and positive effect components, respectively. Negative effect component had a mean score of 2.69. Tension component had 2.68 while competence had 2.29. Lastly, the positive effect component had a mean score of 2.26. The mean score for the immersion scale was found to be 1.89, which could also be considered close to 2.0. This suggests a moderately acceptable level of immersion in the game play. Nonetheless, findings revealed that participants did not have a good experience with the challenge and the flow aspects of the DnD game. The challenge component had a mean score of 1.09 while flow had 1.31. Table 6.8 presents a summary of the game play experience statistics.

The most significant GEQ finding is that participants had positive experiences while playing the generated games on the competence, positive effect, tension, negative effect, and immersion scales. The lowest means were observed in the item challenge for both games, which was also found to be the least reliable of the measures (Mushroom, α=0.67; DnD, α=0.76). These findings provide an opportunity for improvement on the challenge dimension for both games.

Table 6.8 GEQ statistics for the Mushroom Picker and DnD Games, N = 20, GEQ Scale Ranges from 0 to 4 (adapted from [183]

GEQ Components	No. of Items	Means Mushroom	Means DnD
Competence	5	2.11	2.22
Immersion	6	1.94	1.89
Flow	5	1.59	1.31
Tension	3	2.60	2.68
Challenge	5	1.44	1.09
Negative Effect	4	2.61	2.69
Positive Effect	5	2.23	2.26

6.5.2.2 Descriptive statistics

Competence component

Regarding the competence component, the mean scores suggest that students acquired programming skills as a result of playing the given games. In addition, descriptive statistics indicate that students felt that they were skillful, competent and good at it while solving programming tasks. They also noted that they were successful and explored things while playing the Mushroom Picker and the DnD games. This demonstrates the possibility of a positive learning outcome on players. A summary of the descriptive statistics is shown in Figure 6.11.

Tension component

On the tension component, a majority of students indicated that they did not feel annoyed, irritated or frustrated while playing the given games. For each of the three tension items, the percentage of students who reported negatively was less than 50%. Nonetheless, it seems as if the DnD game slightly frustrated some students. Otherwise, the overall absence of tension suggests that the game play experience was enjoyable and fun. This has the potential of reinforcing student motivation [274], attention and immersion during the learning process [81]. This suggests high Quality of Learning Experience (QoLE) [256] by students. See Figure 6.12 for a summary of the descriptive statistics.

Positive and negative effect components

Only few participants reported negative effect of the game experience. Instead, 50% of the total participants agreed that they experienced a positive effect on the played games. These results confirm the absence of negative in-game experiences like frustration, disappointment, irritation,boredom, and anger or post-game experiences such as regret, guilt,disappointment, anger and revenge [203]. For the Mushroom picker and the DnD games respectively, relatively few participants expressed that the games gave them a bad mood (27%;20%) and that they felt bored (25%;27%). This shows that par-

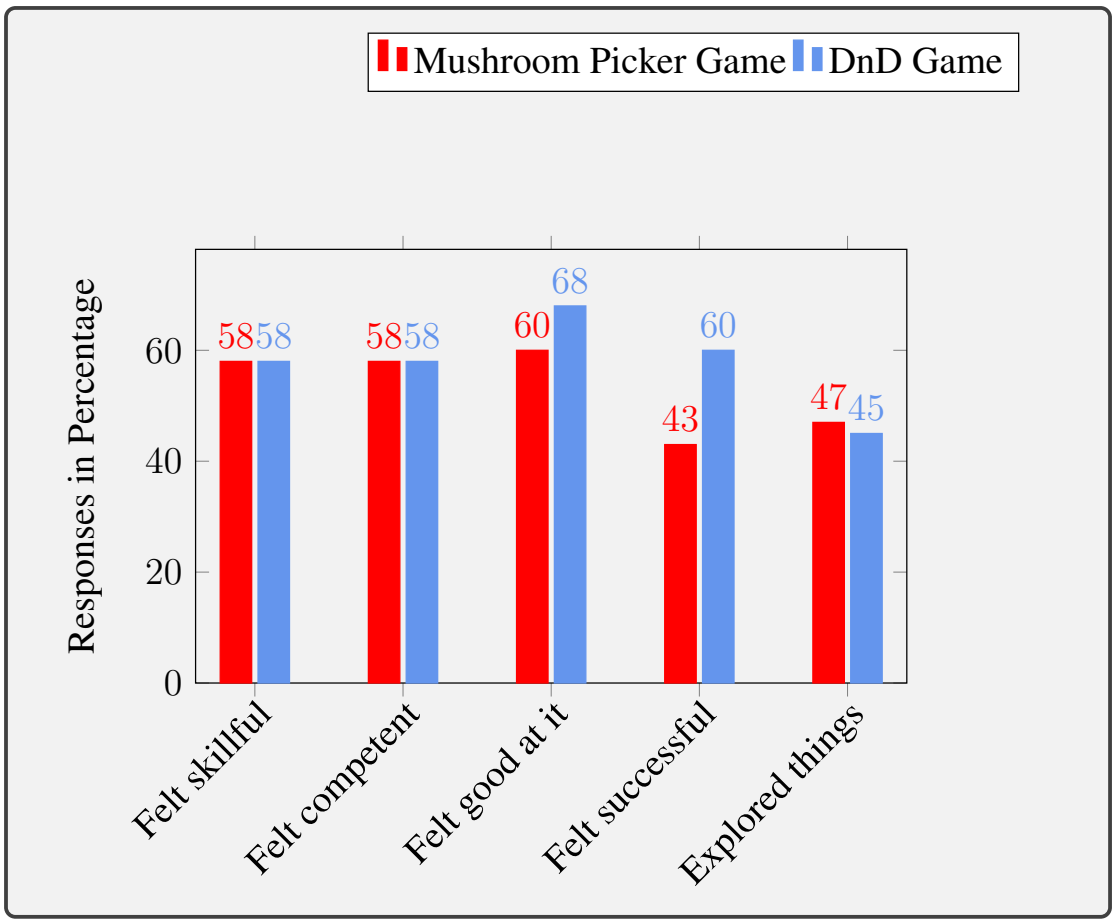


Fig. 6.11 Perceived Competence

ticipants experienced the played games as a positive learning intervention, hence the potential of offering students the opportunity and motivation to overcome the difficulties usually encountered when learning the topic of recursion. However, a slightly high number of participants thought of other things(36%;40%) and found the two games tiresome(45%;45%). Figures 6.13 and 6.14 are visualisations of the descriptive statistics.

Reliability test

To evaluate participants'game play experiences, first, Cronbach's reliability test was conducted. This was done to test the average level of correlation between all the items in each of the Game Experience Questionnaire dimensions [183]. The measure tested inter-item reliability to see if all the items in each dimension or scale measured the same things. An Alpha (α) score of at least 0.70 or higher was taken to indicate a good level of inter-item reliability. For the Mushroom Picker game, most GEQ dimensions showed acceptable levels of reliability (Competence; α = 0.92, Immersion; α=*0.95*, Flow; α=0.84, Tension; α=0.84, Negative effect; α=0.79, and Positive effect; α=0.84). For the DnD game, it was found that all the GEQ scales showed acceptable levels of reliability (Competence; α = 0.88, Immersion; α=0.93, Flow; α=0.79, Tension; α=0.90,

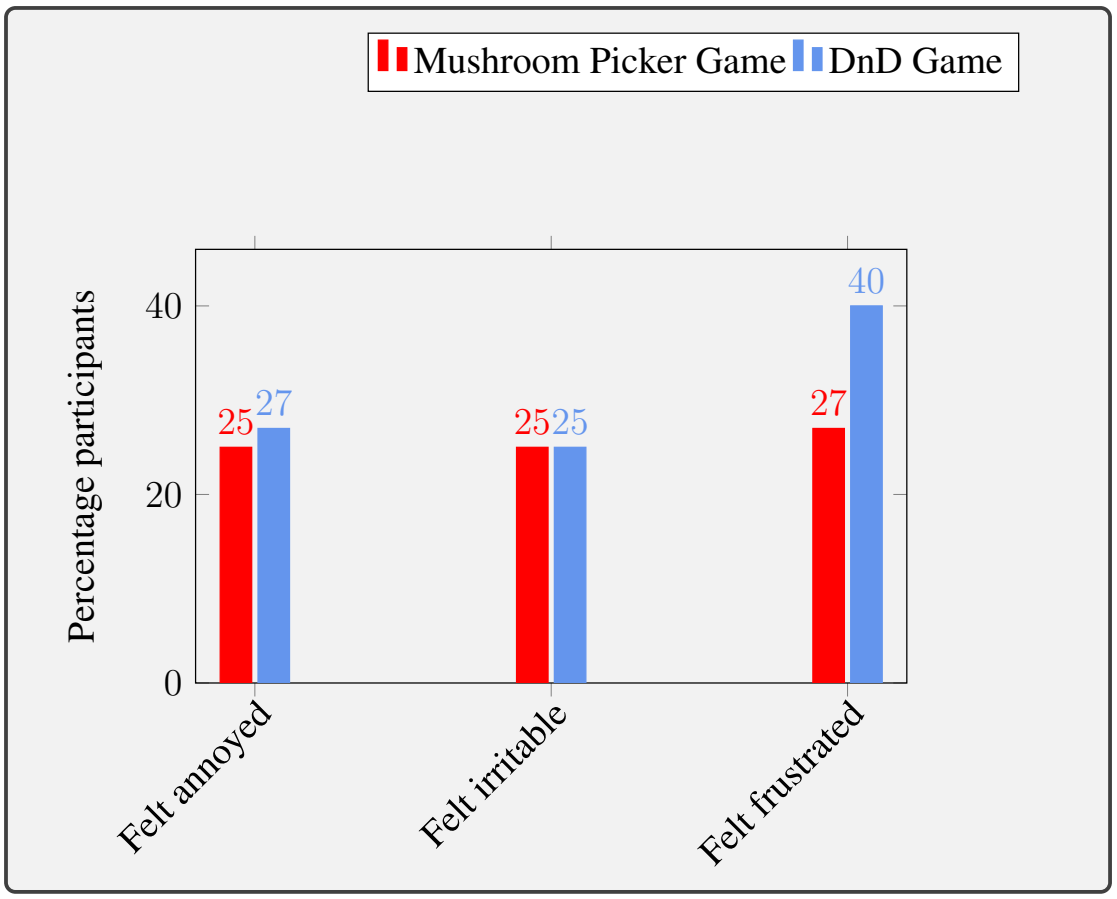


Fig. 6.12 Perceived tension

Challenge; α=0.76, Negative effect; α=0.90, and Positive effect; α=0.93). A summary of the reliability statistics is presented in Table 6.9.

6.5.3 Finding 3: CS1 Students'Perception of Played Games

To gain further insights on the game experiences with the played games, additional items investigated specific design aspects of the played games. Others specifically elicited insights on whether participants found the played games effective for learning the topic of recursion or not. Figure 6.15 is a visualisation of the results.

A significant finding is that 60% of the total participants agreed that they found the played Mushroom Picker game effective for learning the concept of recursion. In addition, another 60% of the total participants reported that they found the visualisation feature in the Mushroom Picker game a better way of learning the concept of recursion. At the same time, approximately another 50% of the total participants found practicing coding using a Python IDE in the Mushroom game environment a motivating way of learning recursion. 30% disagreed that the IDE was motivating whereas 20% were neutral. Nonetheless, considering the findings on the ranking of the most useful design features of the played games (See Table 6.10), generally the IDE feature was considered

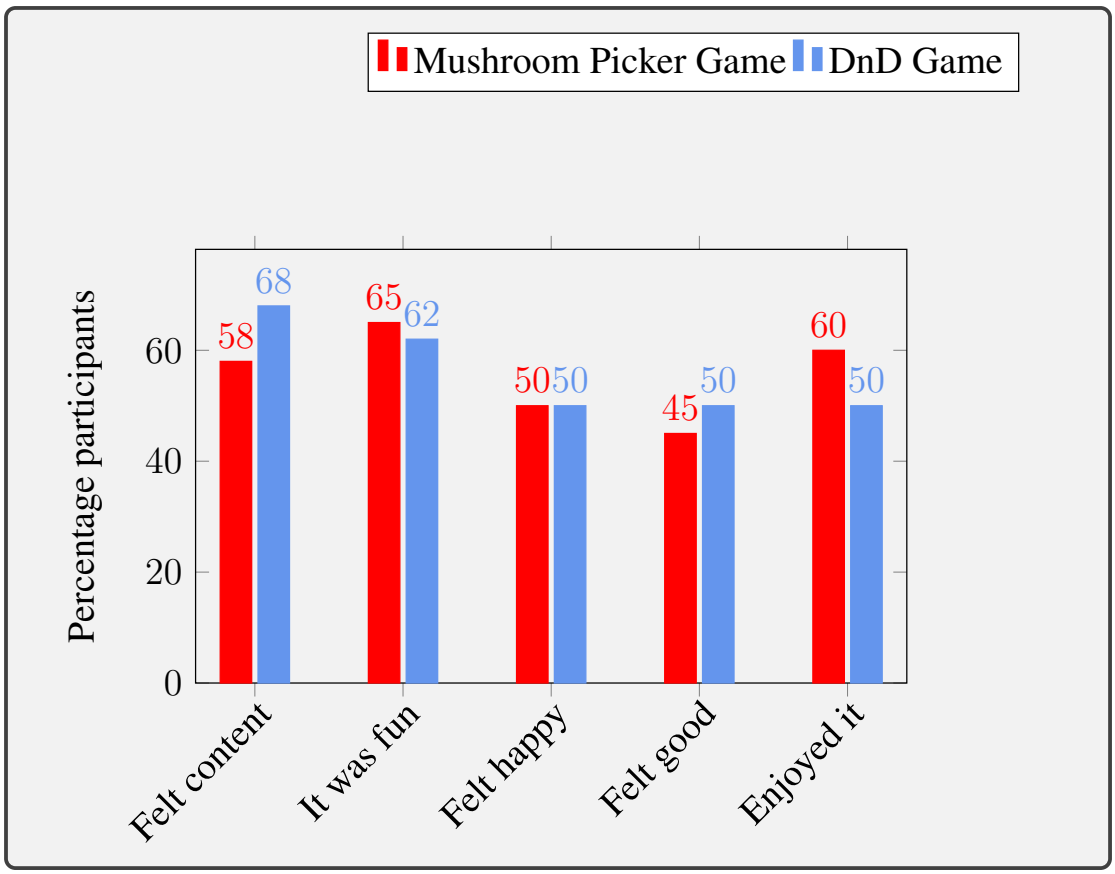


Fig. 6.13 Perceived Positive Effect

important. Half of the participants expressed that the mushroom analogy chosen for the Mushroom Picker game was a better way of learning recursion. Generally, these results demonstrate the educational potential of this game's design for learning programming among CS1 students.

6.5.4 Finding 4: Rankings of most useful design features

When participants were asked to rank the most useful design features of the played games, results indicated visualisation (85%), programming learning tasks (80%), Python IDE (75%), player reward and motivation (50%), and navigation and interactions (35%). At the same time, the least ranked features were color and design layout (30%) and both multimedia elements and character customisation at 25%. This is almost the opposite of teacher responses. The fact that students cannot customise the games could be the reason why. Table 6.10 is a summary of the games design features'rankings.

6.5.5 Open ended questions responses

Finally, two optional open ended questions were asked to participants to gain more insights that could further evaluate the effectiveness of the proposed game generator

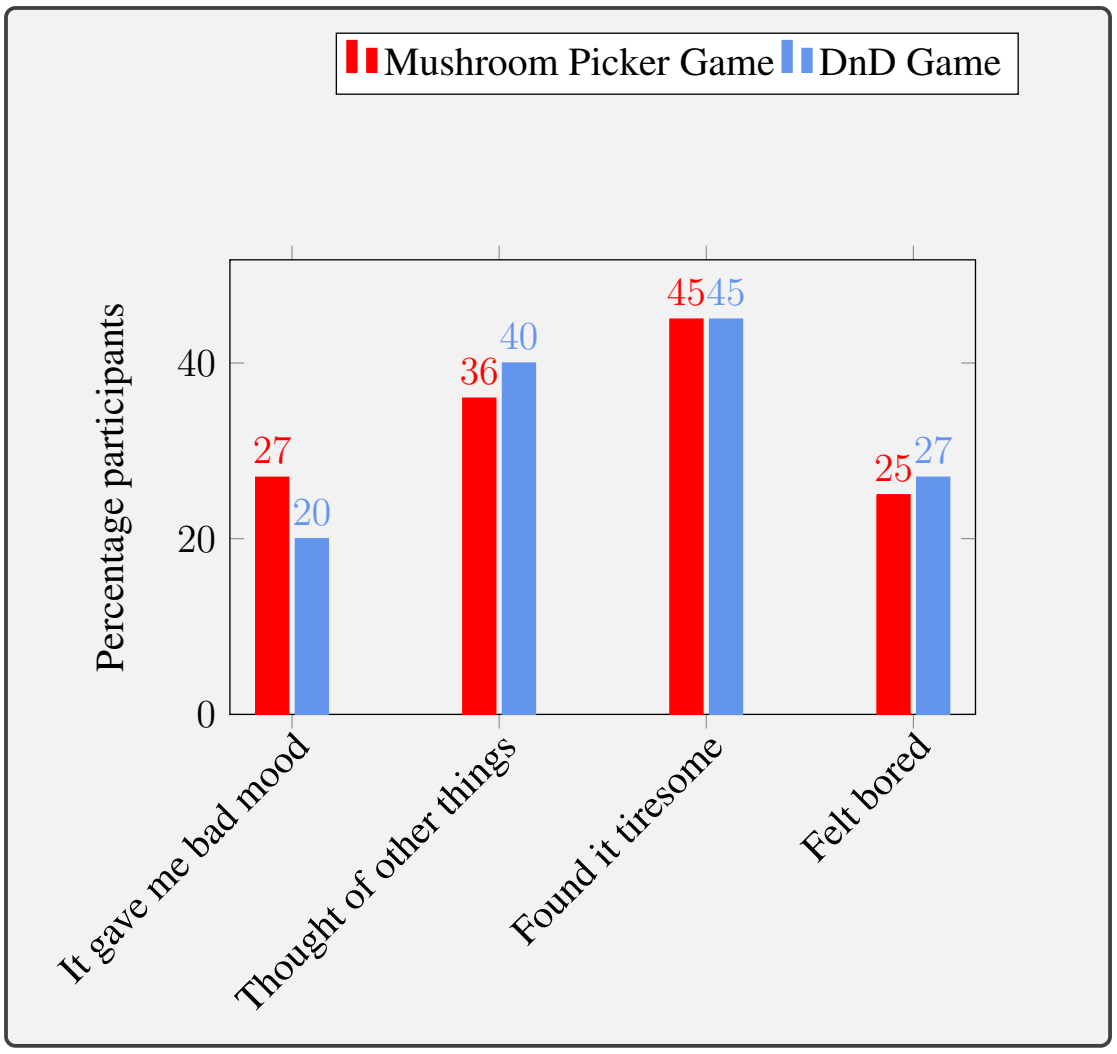


Fig. 6.14 Perceived Negative Effect

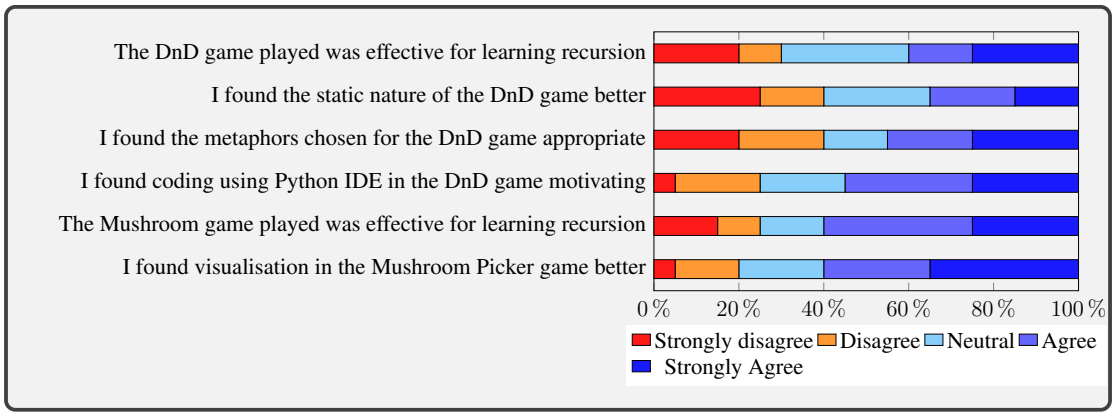


Fig. 6.15 Students'perceptions of played games on additional items

tool concept from the perspective of students. The first question read - '*what is your final comment about the played programming games* '? Nineteen participants responded to this question. In their own words, most of them reported that the played games

Table 6.9 Reliability statistics for the Mushroom Picker and DnD Games, N=20

GEQ Components	No. of Items	Cronbach's α Mushroom	Cronbach's α DnD
Competence	5	0.92	0.88
Immersion	6	0.95	0.93
Flow	5	0.84	0.79
Tension	3	0.84	0.90
Challenge	5	0.67	0.76
Negative Effect	4	0.79	0.90
Positive Effect	5	0.84	0.93

Table 6.10 Ranking of most useful design features of played games

Feature	Participants↓	Percentage↓
Visualisation	17	85%
Programming learning tasks	16	80%
Python Integrated Development Environment	15	75%
Player reward and motivation	10	50%
Navigation and interactions	7	35%
Color use and design layout	6	30%
Multimedia elements	5	25%
Character customisation	5	25%

were effective and had great potential of enabling students to learn the difficult topic of recursion.

By giving a personal experience when first taught the recursion topic, one student expressed how difficult it was to understand but acknowledged that the generated games made it easier. This is captured in the excerpt: *"When I was initially taught Recursion in CSC1015F, I found it extremely hard to understand due to the fact it was pretty unconventional compared to standard looping mechanisms... however, in the way that the Mushroom Picker game was presented, I felt it was much easier to grasp the concepts being taught....furthermore, the gradual approach of the games was extremely effective in comparison to being taught the entire section in a few days with few physical examples*'.

To further illustrate the perceived effectiveness of the proposed game generator tool, we performed a detailed thematic content analysis. Four themes emerged, namely: (i) easier/ simple; (ii) potential; (iii) help; and (iv) recommend. Table 6.11 presents a detailed thematic analysis.

Another significant result was that students preferred learning programming using games in future and recommended the adoption of GBL in teaching and learning programming. This can be illustrated by two responses: (i) "*this games are way too good*

Table 6.11 Detailed thematic analysis of RGG's effectiveness and potential

Theme	Aspect	Sample quotations
Easier	Generated games made learning recursion easier/ simpler	"*I think it helped me understand the concept better and it was easier to understand through visuals and working through a game*". "*They were quite helpful to a large degree and make learning programming a lot easier*". "*It helped a lot in learning programming as it made it a bit simpler to learn*".
Potential	Generated games have the potential to teach recursion	"*The mushroom picker game, if combined with story like the DnD game, has the most potential to be useful as a tool for learning recursion*". "*I think there is potential for using games to teach concepts like recursion*".
Helped	Games helped students learn programming	"*They were quite helpful to a large degree.....*". "*It helped a lot in learning programming*".
Fun	Generated games are fun, enjoyable and can motivate students	"*the games were very fun and provided a much more enjoyable and different learning experience which I think should be adopted here and there*". "*I enjoyed the games, especially the DnD game....adventure is a good genre for learning and it matched the concept (recursion) nicely...I look forward to another more games*".

and I think it's time kids play those kind of games to learn programming", and (ii) "*I think the concept is cool and it would be nice to learn recursion using interactive games*"

In a surprising finding, one student expressed unhappiness with the level of challenge in the Mushroom Picker game, noting that it was more suited for a novice. The participant nonetheless liked the challenge in the DnD game. This can be understood from the quote: *"I disliked the Mushroom game and think it's too much for a novice but I enjoy the DnD game with its explanations and bite sized code...it really made recursion easier, when it's taught step by step like that....I could see myself getting into coding as a hobby if it was taught with games"*. The revelation in the last part of this feedback further demonstrates the potential of the played games to positively influence the attitudes of students towards programming.

One student who presumably was not very competent in programming said *"they were a bit difficult to understand...it was hard to determine what to do taking into account that I don't know recursion"*. This statement when analysed alongside the feeling that the Mushroom game was more designed for novices clearly confirms the challenge

of designing programming games for the mixed abilities exhibited in most CS1 classes. Some students seemed to appreciate the visual aspect of the Mushroom Picker game. To illustrate this, some direct statements follow:

- *"It was beneficial with regards to the visualisation of concepts relating to recursion "*.
- *"I think it helped me understand the concept better and it was easier to understand through visuals and working through a game."*.
- *"I think that the fact that it was so interactive and visual is what helped me better understand the concepts of recursion"*.

Two students seemed frustrated with the games. The first did not see any educational value in them. He said, '*the games did not actually teach anything ..they were more of a test of my skills..there was no information or way to learn about recursion*'. On the other hand, the second acknowledged that the recursion concept was effectively implemented in the games but noted that the layout was confusing, the instructions were poorly worded and that the overall frustrating UX made it difficult to get to the parts of the games that actually worked.

The second question was - '*what changes or improvements would you like to see in the played games to improve their effectiveness for learning programming* '? The aim of this question was to elicit views of participants on design aspects of the played games that needed improvement. Mixed reactions were reported on the suggested improvements for the played DnD and the Mushroom Picker games. Students proposed improvements on the: (i) analogy used in the DnD game; (ii) IDE and interface design of the played games; (iii) target topics so as to include other programming concepts in addition to recursion; (iv) games'level of challenge; (v) user guides and tutorials; (vi) DnD game visualisation; and (vii) feedback mechanisms such as use of leader-boards.

One student felt that the analogy used for the DnD game should have been improved. The student expressed: *"I do not believe that the analogy used for the D and D game was effective. If compared to the Mushroom Picker game, where there was a clear set of steps that needed to be repeated, the DnD game was harder to visualize. I think that a more fitting analogy would greatly improve the learning experience of the D and D game. For example, something like a car being required to move around a race track for a specified number of laps would have been more fitting"*.

Some students reflected that the IDE and interface design of the played games ought to have been improved as well as the help feature. In particular, that of the DnD game. This is demonstrated by the excerpt: *"I think the tutorials should be more in depth. I really struggled with the Mushroom Game. To me the first level's code was too complex and I struggled. The IDE was also not ideal in that game. It was however very suited to*

the DnD game. The DnD game had me whizz through the levels because the explanations and manageable goals really helped". Students also felt that the generated games should not be limited to teaching only the recursion topic but should include other CS1 topics as well. This is illustrated by response", *"do not limit the games to recursion, teach almost every module using games or real world application"*.

While a majority of students acknowledged the suitability of the played games for learning recursion, some noted that they wanted more challenge, steady build up of tasks, clear tasks, good feedback and help in the form of tutorials. This can be demonstrated from the excerpts: (i) *"The games are fine for learning, though I felt I wanted more.....a little more challenge would be nice or rather extra stages"*; (ii) *"There should be more of a build up as in DnD to a whole recursion solution...I also feel as though the tasks were not very clearly and it would be nicer to get some sort of feedback when things go wrong otherwise it is frustrating. Maybe a leader-board could be nice to motivate people too....The DnD game was very nice to play to learn recursion as it asked for tasks that made sense and there was a gradual build up to a full recursive call that we had to do ourselves"*; and (iii) *"I think the tutorials should be more in depth. I really struggled with the Mushroom Game. To me the first level's code was too complex and I struggled. The IDE was also not ideal in that game. It was however very suited to the DnD game. The DnD game had me whizz through the levels because the explanations and manageable goals really helped"*. Raw data on final student comments can be found in Appendix C14.

6.5.5.1 Discussion

The game experience questionnaire revealed that it accurately measured its components, except for the challenge component in the Mushroom picker game. Challenge recorded a slightly lower Cronbach's alpha value (0.67) than the expected score of 0.70 or above. Game experience findings show that participants had a good experience while playing the given generated games, in particular, on the competence, positive effect, negative effect, tension, and immersion dimensions. Results further indicate that different students had mixed experiences with the challenge designed in the played games. For example, whereas some liked the gradual introduction of challenge in the Mushroom Picker game, others preferred that of the DnD game. This confirms that the played games met the need to design learning interventions to cater for diversity in higher education [51, 25, 222].

In addition, as was the finding in the third experiment with CS1 trainees, it was confirmed that the visual and interactive aspects of the Mushroom Picker game proved to be an effective pedagogical approach of teaching programming [73, 48, 186, 197, 278]. Moreover, the feedback that a majority of students found the games fun and

enjoyable demonstrates their potential to engage, motivate and retain students in CS1 [81].

As confirmed by a response from one of the participants when giving final comments, the played games seemed to have changed the attitudes [274] of some students towards programming. There is also evidence to suggest that the approach could be adopted to teach other programming topics in CS1 other than the recursion concept. Finally, the finding that the supported IDE feature in the game environment was one of the highly ranked indicates that design could enable players to gain knowledge through exploration and practice [61]. The finding is in line with the constructivist design principle of programming games [48].

Regarding the relatively large percentages of participants who chose to remain neutral on some items, we predict that this could be explained by fatigue on the side of participants. Respondents could have been involved in more than one online survey given that most user studies at the University were being conducted online due to the Covid-19 pandemic. Another reason could have been that may be they were not as impressed with a game as their teachers.

One possible threat to validity of the findings could be the slightly low number of participants. Nonetheless, an effort was made to attract both weak and strong programming students through monetary compensation. In addition, this sample size is in line with those of similar prior evaluation studies of game authoring tools with students (n=20) [76, 150]. Therefore, overall, it could be argued that the empirical findings of this experiment suggests that the games generated by the prototype game generator tool called RGG had an educational potential from the lens of students. They could enable CS1 students to learn the difficult topic of recursion and possibly other abstract CS1 topics in higher education. The student experiment was supposed to make a different contribution from those of the teacher and trainee experiments.

6.5.6 Summary

This Chapter presented results obtained from four evaluation studies carried out in this thesis. The first 3 evaluated the proposed game generator tool with teachers. The last (experiment four), evaluated the generated games with students. In the first experiment, the prototype was evaluated with 30 CS1 instructors from Kenya and South Africa who answered a survey on the usability. Results demonstrated that the prototype is effective, efficient, useful, and easy to use and learn. Additionally, instructors were satisfied with the prototype and highly recommend adoption of a game generator tool idea for teaching programming.

In the second experiment, the prototype was evaluated with 28 CS1 instructors from seven countries who answered a survey on perceived usefulness. Results show that the

prototype is useful. Generally, the top five most useful features suggest that the prototype could significantly reduce the effort of game generation. Therefore, it could be argued that the findings provide additional empirical evidence of the prototype's usefulness, demonstrating the adoption potential of game authoring platforms as support tools in CSE. In addition, the CSE community could consider the identified useful design features to inform design of other game based learning tools.

The third experiment evaluated the prototype with 22 CS trainee teachers from Kenyatta University. Evidence gathered established that respondents had a good UX with the prototype and found the game generator idea to be good. This thesis therefore presents the argument that supporting CS trainee teachers with such game generator platforms has the potential of advancing the adoption of GBL in CSE education, particularly among this audience. The proposed idea of a programming game authoring tool could benefit both high school and higher education (tertiary) CS teachers. Lastly, experiment 4 evaluated the prototype by assessing the game play experience of students. Results indicate that students had a positive game play experience with the generated games, suggesting a positive learning impact on players.

Overall, the positive experiences of participants in the four evaluation experiments suggest that such a higher-order tool has the potential to effectively support teachers and trainees to easily create customised serious games and that the generated games can effectively help students to learn programming. Consequently, the tool could broadly meet the needs of a diverse audience of instructors and students, hence increase the adoption of serious games in programming education. As such, the second research question is answered.

6.6 Comparative analysis of participant cohorts results

It is worth noting that teachers also evaluated and commented on the generated games. When results from the four evaluation experiments are analysed comparatively across the three participant cohorts, there are some comparisons and contrasts. For example, teachers, trainees and students found the visualisation feature and the generated games useful for supporting teaching and learning programming, particularly, the recursion topic. All the three cohorts found the Mushroom-Picker game effective for learning recursion compared to the DnD game. Moreover, they were all in agreement that the generated games can apply to other CS1 topics. In contrast, not all the participant cohorts found the IDE feature useful. Results indicate that only students and trainees felt that the IDE feature was useful. It was surprising that teachers did not consider this feature useful. The reason could be that their evaluation was more focused on the game generator tool as confirmed by their positive ratings of the prototype's usability, novelty and the fact that it has a built-in game generation template.

Meanwhile, only teachers and trainees found the customisation feature useful. On the other hand, only trainees and students thought that the reward/ motivation feature was important but not teachers. These findings could be explained by the assumption that teachers and trainees were mostly interested in a toolkit that could support customisable game generation, whereas students were keen on a fun environment for learning programming. A summary of the comparative analysis of the findings from the 4 evaluation studies is shown in Table 6.12.

Table 6.12 Comparative Analysis of Participant Cohorts Results

Aspect	**CS Teachers**	**Trainee Teachers**	**CS1 Students**
Found visualisation feature useful	✓	✓	✓
Were impressed with the generated games	✓	✓	✓
Would recommend the tool for teaching programming	✓	✓	N/A
Would recommend the generated games for teaching and learning programming	✓	✓	✓
Generated games can support learning programming	✓	✓	✓
Generated games can apply to other CS1 topics	✓	✓	✓
Generated games are useful for teaching and learning recursion	✓	✓	✓
Found customisation useful	✓	✓	×
Found IDE feature useful	×	✓	✓
Found help feature useful	×	×	×
Found reward/ motivation feature useful	×	✓	✓
Found RGG tool usable	✓	✓	N/A
Thought game generation was a novel idea	✓	✓	N/A
Found built-in game generation template useful	✓	✓	N/A
Found Mushroom-Picker game effective for learning recursion	✓	✓	✓
Found DnD game effective for learning recursion	×	×	×

CHAPTER 7

Conclusions

This Chapter concludes the work done in this research study. Section 7.1 presents a summary of the thesis. Section 7.2 is a summary of the study findings against research questions. Section 7.3 presents the research implications. This is followed by research limitations in Section 7.4. Lastly, Section 7.5 gives future research and final reflection.

7.1 Summary of thesis

The use of games in education is not new. Serious games have the potential to motivate and engage students. As a way of promoting the adoption of GBL in programming education, this thesis investigated a game generator tool to support non-technical educators to easily create customised games. The work presented in this thesis was organised as follows:

Chapter 1 introduces the study by providing a brief background, stating the problem, research statement, and research questions. It also gives an overview of the research scope, design, and contributions.

Chapter 2 gives a comprehensive review of past works related to the topic under investigation.

Chapter 3 discusses three studies conducted to understand the user needs and requirements to inform the design of a prototype game generator tool.

Chapter 4 presents the design and development of the prototype game generator tool called RGG and provides justification for design choices.

Chapter 5 is on the experimental design followed when evaluating the prototype game generator tool.

Chapter 6 presents the results obtained from four empirical studies done to evaluate the prototype game generator tool and the generated games.

Chapter 7, the last in this thesis, concludes the research study by summarising findings against the research question, implications, limitations and future work.

7.2 Summary of the study findings against research questions.

This study aimed to investigate a game generator tool to support teachers with little or no game programming skills to easily create games to teach recursion in higher education. Results indicate that CS educators found the designed prototype (RGG) effective for creating programming games to teach recursion. Further results show that CS1 students also found the generated games effective and suitable for learning the topic of recursion. Broadly, this thesis sought to answer three research questions.

RQ 1: What conceptual attributes must a generator tool take into account in order to generate a game that can teach recursion?

RQ 2: How effective is the use of a generator tool in creating customised game instances to teach recursion?

RQ 3: What is the suitability and educational potential of the generated games to enable students to learn the topic of recursion?

7.2.1 RQ 1: What conceptual attributes must a generator tool take into account in order to generate a game that can teach recursion?

In order to address RQ 1, three sub-questions were further raised and answered. These sub-questions were significant as they guided the design of a prototype game generator tool that was used in the evaluation experiments.

Sub Qn 1: What teaching practices are used by programming teachers in Kenya and South Africa when teaching undergraduate novice students the recursion topic?

In order to understand the conceptual design attributes, it was deemed necessary to have insights on the current teaching practices adopted by educators when teaching the recursion topic. It was found that higher education teachers from Kenya and South Africa spend little time on the topic. In addition, most of them use mathematical text book examples such as the Factorial, Fibonacci series, GCD, binary search, power of a number, palindrome, strings, towers of Hanoi, and recursive patterns. Little use is made of real life scenarios.

Sub Qn 2: What teaching examples or analogies are suitable for designing games to teach recursion to diverse novice students from Kenya and South Africa?

Given their teaching experience, it was deemed that teachers would provide useful information on some examples and analogies they thought would be useful to teach diverse novice students recursion. Results indicate that inclusivity in programming games design could be advanced by using a variety of teaching examples that could cater to different learning needs. In particular, simple text book examples and real life analogies could be designed, bearing in mind individual learning differences among

learners. It was also observed that care must be taken to ensure that the analogies used do not introduce extra cognitive load on students.

Sub Qn 3: What design considerations are useful when developing a game authoring tool to create games that can teach diverse novice students recursion in Kenya and South Africa ?

To answer this sub-question, three aspects were examined: (i) unique design factors; (ii) customisable game attributes; and (iii) pedagogy. Results indicate that the unique design factors that ought to be considered are context, gender, religion, and childhood game experience. Meanwhile, the customisable game attributes that teachers could customise when authoring different instances of recursion games include game genre, scenario, background, character, interaction style and complexity. Finally, with regard to pedagogy, it was found that there is need to design instructions in a way that balances game play and learning. The design should: (i) consider individual differences among learners, (ii) support construction of knowledge by enabling learners to practice coding in the game environment, (iii) clearly align game goals and learning goals, (iv) ensure player fun, engagement and motivation, and (v) promote scaffolding and student help.

In an attempt to answer the first research question, this thesis proposes eight conceptual design guidelines/ principles that could guide requirements engineering and creation of diverse games to teach diverse students programming. This is a significant contribution to the body of knowledge in educational games development given the lack of serious games design principles [52]. Details are discussed in Chapter 3, Section 3.3.

7.2.2 RQ 2: How effective is the use of a generator tool in creating customised game instances to teach recursion

To address the second research question and to demonstrate the effectiveness of the proposed game generator tool, three evaluation experiments were conducted. For each, additional sub-questions were posed and answered as follows;

Experiment One

Sub Qn 1: How useful and easy to use do programming instructors from Kenya and South Africa find the RGG generator tool?

Overall, most CS1 instructors from Kenya and South Africa found the tool useful. In this regard, a majority of teachers agreed that the tool would: (i) make them be more effective and productive, (ii) be useful in creating CS1 games, (iii) give them more control, (iv) make things easier to get done and, (v) save them time. Teachers reported that the prototype tool was easy to use, simple to use, user friendly and required fewest steps to accomplish the tasks they wanted done. In addition, all post task ratings were positive. For example, participants did not find game generation tasks difficult. As a measure of task efficiency, an estimate of the time spent while performing game

generation tasks demonstrates that most participants did not spend much time. Lastly, most participants were successful in performing the given game generation tasks.

Sub Qn 2: What is the opinion of programming instructors from Kenya and South Africa about the RGG tool's learnability?

Findings of the first experiment indicate that most participants found the tool easy to learn. Particularly, they noted that they learnt the tool quickly and could easily remember how to use it.

Sub Qn 3: How satisfied are programming instructors from Kenya and South Africa with the RGG generator tool?

Regarding the satisfaction metric, CS1 instructors from Kenya and South Africa reported that they were satisfied with the prototype game generator tool. They saw an opportunity in the concept of a game generator tool to support CS educators in their daily work. They also observed that the generated games would be useful for students to learn the difficult topic of recursion. In addition, they said that they would recommend the prototype game generator tool to fellow programming teachers.

Overall, results of the usability evaluation of the prototype game generator tool with CS1 teachers demonstrate that the tool could be effective in supporting CS instructors in creating games to teach novices the recursion topic. Similarly, this could be generalised to other abstract CS1 topics.

Experiment Two

Sub Qn 1: How useful do CS1 instructors from the wider CSE community find the prototype game authoring tool after using it to create games to teach recursion?

CS1 instructors from the wider CS education community found the prototype game generator tool useful. A majority of participants (75%) thought that the prototype could be applied in many other CS1 topics and that the generated games are easy to distribute. Another 72% noted that it could increase integration of the educational value and that the generated games could support student centered learning. It was also reported by 71% of the total participants that the prototype could reduce game production cost and effort. Finally, 68% of the total participants thought that the prototype is useful to both students and teachers and that the generated games are useful for teaching the recursion topic in CS1.

Sub Qn 2: What design features do CS1 instructors from the wider CSE community find most useful?

The top five most useful design features are custom game generation, built-in game generation template, support for common operating systems, no game programming required, game download and distribution and IDE, respectively.

Experiment Three

Sub Qn 1: What is the adoption potential of a support tool that uses the concept of a game generator to help CS trainee teachers to create games that can teach programming?

Mean values for the four AttrakDiff dimensions indicate that CS1 trainee teachers had a positive user experience with the prototype, suggesting a high probability of adoption. In additional, qualitative comments showed a positive user experience, further demonstrating the tool's potential educational value and adoption.

Sub Qn 2: What is the overall opinion of CS trainee teachers about RGG and the generated games?

CS trainee teachers see the prototype game generator (RGG) as a useful, easy to use and learn toolkit for supporting trainees who may wish to adopt GBL but find it a barrier due to limited game programming skills. In additional, they see great potential in the generated games to enable students to learn the topic of recursion.

7.2.3 RQ 3: What is the suitability and educational potential of the generated games to enable students to learn the topic of recursion?

In order to answer the third research question, CS1 students played and evaluated the generated games and gave feedback on their suitability and educational potential to learn the recursion topic. This was done in experiment four.

Experiment Four

In this experiment, three sub-questions were answered as follows:

Sub Qn 1: What is the experience of CS1 students after playing games created from a prototype game generator tool called RGG to learn recursion?

Students'feedback on the Game Experience Questionnaire (GEQ) show that they had positive experiences while playing the generated games. The positive experiences were reported on the competence, positive effect, tension, negative effect, and immersion scales.

Sub Qn 2: Which game design features did CS1 students find most useful?

The five most useful game design features according to students were visualisation (85%), programming learning tasks (80%), Python IDE (75%), player reward and motivation (50%), and navigation and interactions (35%). Students also found the analogies used in the generated games appropriate for learning the topic of recursion.

Sub Qn 3: What is the overall opinion of CS1 students about the potential of the generated games to support students to learn the topic of recursion?

In general, CS1 students see great potential in the generated games and find them useful to enable students to learn the topic of recursion. They mention that the played games helped them to better understand, made it easy or simpler to learn recursion, and

helped them visualize the difficult topic. Furthermore, the played games were considered fun and enjoyable. Finally, students recommend the adoption of such games to teach the recursion topic and other programming concepts. Lastly, one striking finding from one of the students points to the fact that the played games could potentially change the attitudes of students towards programming.

Overall, in response to RQ 2: the evaluation by CS teachers found the prototype useful, easy to use and learn and participants said that they were satisfied with it. Teachers expressed a good user experience. Meanwhile, trainee teacher responses to the Attrackdiff 2 questionnaire also indicate a positive user experience. In response to RQ 3, CS1 students'feedback on their game play experiences are mostly positive, indicating the educational potential of the generated games. Consequently, findings of the 4 evaluation experiments provide empirical evidence to demonstrate the effectiveness of the prototype game generator tool and the educational value of the generated games. Table 7.1 is a summary of the four empirical evaluations.

Table 7.1 Summary of of Empirical Evaluation studies

	Experiments 1 and 2	**Experiment 3**	**Experiment 4**
Aim	To evaluate the usability and user experience of CS educators when using RGG to create games to teach recursion	To evaluate the usability and user experiences of trainee teachers when using the RGG to create games to teach recursion	To evaluate the educational potential of the games generated using RGG to learn recursion.
Participants	CS Teachers (N=30); and(N=28)	Trainee teachers (N=22)	CS1 students (N=20)
Study context	Online studies	Controlled lab experiment	Online study
Instruments	USE Questionnaire	Attrakdiff 2 Questionnaire	Game Experience Questionnaire
Evaluation criteria	Usability, user experience, perceived task performance, and perceived adoption potential	Usability, user experience and perceived adoption potential	Game play experience and perceived educational potential of the generated games
Evaluation metrics	Perceived usefulness, ease of use, learnability, user satisfaction, and efficiency	Hedonic Quality Identity, Hedonic Quality Stimulation,Pragmatic Quality, Attractiveness, and perceived tool's adoption potential	Perceived immersion, flow, competence, negative effect, positive effect, tension, challenge, and educational potential of the generated games
Results	Teachers found the prototype usable, had a good User Experience and highly recommended the adoption in teaching programming - demonstrating its effectiveness	Trainee teachers found the prototype usable, had a good user experience and highly recommended the adoption in teaching programming - demonstrating its effectiveness	Students had a positive game play experience and expressed the educational potential of the generated games in learning programming - demonstrating its effectiveness
Contribution	Empirical evidence of the usability of a prototype game generator tool by CS1 educators	Empirical evidence of the potential and suitability of a prototype game generator tool to support CS trainee teachers	Empirical evidence of the effectiveness and suitability of the generated games to enable CS1 students to learn the difficult recursion topic

7.3 Research Implications

The results of this research have possible theoretical and practical underpinnings for the design of serious games to teach programming with regard to pedagogy.

7.3.1 Theoretical Implication

The topic on diversity and inclusivity in serious games design featured prominently in Chapter 3. In this thesis, results of the requirements analysis study show that when designing programming games, consideration ought to be given to diversity. This is consistent with findings from the work by Mohamed [176]. As a theoretical contribution and to advance research in this area, this thesis proposes eight conceptual design principles that could guide the development of serious games to teach novices programming [18] as discussed in Chapter 3. These principles could be generalised to the design of other similar GBL learning interventions meant for CS students.

7.3.2 Practical implications

7.3.2.1 *Visual coding*

This research indicates that novices found learning the topic of recursion through visualisation effective. This finding is consistent with those of some previous works [148, 171, 207]. The implication of this on educational programming games design is to give players (students) an environment where they can visualise the execution of their codes or algorithms. It also suggests that educators should design instructional tasks that encourage visualisations, especially for abstract programming tasks.

7.3.2.2 *Constructivism in design*

Results of the research also show that students found practicing coding using a built-in IDE in the game-play environment a better and useful way to learn programming. This has been shown to encourage active involvement and participation in the learning process [38]. As such, students could construct their own knowledge [75, 121]. The design implication for serious games to teach programming is to consider having an IDE embedded in the game-play environment. This could allow students to practice coding in a game-full and more engaging environment [26].

7.3.2.3 *Motivation and fun*

Serious games are well known for their motivation, fun and retention benefits to students [31, 250, 275]. In this research, findings suggest that the design of reward systems in the form of health points and trophies could motivate students. The implication for the

design of programming games is to ensure that players are entertained while they learn or acquire the targeted skills. Given the difficult nature of programming, this design aspect could be very useful if learners are to be retained in CS courses, especially CS1 [26, 214].

7.3.2.4 *Scaffolding and learner help*

Novice learners often need support as they gradually learn how to program [157]. In the current research, it was found that scaffolding and student help are among the top most useful aspects to students. In literature, Malliarakis et al. and Barness et al. echoed the need to consider scaffolding in the design of programming games [26, 148]. Likewise, this research confirms that it is worth considering scaffolding when designing programming games in higher education.

7.3.2.5 *Innovation and creativity*

A recent study identified a research gap in creativity in CSE [237], particularly in CS1. Creativity in CS involves the '*generation of adaptive, useful solutions to problems that are novel within the relevant context*' [237, p. 4]. The authors argue that one example of creativity and innovation required in CS could be game development projects, among others [237]. To address this gap, this thesis proposes to the CSE community a practical, innovative and a novel game generator tool to support CS educators [19, 16]. This is one of the research outputs of this thesis. The games generated by the prototype game generator tool could also inspire creative thinking among students in the practice of learning programming [237].

7.4 Research Limitations

Despite the enormous potential of this research, it was not without some limitations. The following are some.

7.4.1 Scope

With regard to scope, currently, the generated games can only be played on Windows and Linux operating systems (OSs). In addition, due to time constraints, audio effects were not designed in the games generated by the current prototype. Another limitation of this research is the few levels designed in the generated games. However, the designed levels are considered reasonable to test the game generation concept. In addition, this research was only limited to the topic of recursion as a case study. However, the results of this study are generalisable as the prototype can not only generate games

to teach the recursion topic but also other CS1 topics as was observed by some participants. This is made possible by a built-in game generator template. Moreover, the proposed game generator approach could be instantiated in many CSE courses other than CS1. The built-in template is general and independent of the recursion topic or CS1 course. Thus, it could be applied to generate other instances of educational games for other CS1 topics or CSE courses.

7.4.2 Sample size

The small number of participants in various studies conducted in this thesis could be a potential threat to validity and generalisability of results. Another limitation could be the small number of participants who had used games in teaching. Nonetheless, it should be noted that some studies have equally shown that lack of educational game authoring tools [252] and other potential barriers [30] have resulted in low GBL adoption in mainstream teaching, particularly in developing countries. Therefore, the small number is a clear reflection of this population. Furthermore, for the evaluation study with CS1 students, the sample size is in line with those of similar prior studies (N=20) [76, 150].

7.4.3 Student learning gain

Lastly, evaluation of student learning gain was not done in this thesis. The reason for this was that there are many confounding variables that could have influenced the student learning gain. Consequently, it would be difficult to conclude that any improvement or gain observed in student learning was directly attributed to the played games. Furthermore, such a study would need a longer period of time than was feasible in this thesis. Probably, an in-person CS1 course offering would have been ideal for this kind of study. Nonetheless, a game experience evaluation conducted with CS1 students provided early evidence to demonstrate the suitability and educational potential of the generated games from the lens of learners. In addition, this research is about the generator not the game.

7.5 Future Research and Reflection

7.5.1 Future Research

i) Generalised results

In this thesis, the evaluation experiments were mostly conducted with participants from Africa with a few from other parts of the world. While the results provide proof of concept and reasonably demonstrate the effectiveness of using the proposed game generator toolkit by teachers to create custom games, it would be interesting to consider more generalised global experiments. It is believed that diversity was to a greater extent factored in the prototype design as demonstrated in Chapter 3. The experiments also to some extent showed how the proposed tool could adapt to diverse student body. However, more studies could be conducted with more students globally on this aspect.

ii) Individualised attention versus usable tools

Whereas students always want individual attention, teachers prefer to teach the same concepts using usable tools. Consequently, the design of game generator tools should consider both users. Indeed, within the CSE community, research has shown that students feel a strong sense of ownership, originality, and pride even if given a limited degree to create and customise their own games [231]. Consequently, another potential direction for future research could be to investigate individualised student attention and usable game generator tools based on the degree of game customisation.

iii) Controlled Experiments in Online Settings

Given the experience of COVID 19 pandemic, many universities in Kenya were forced to migrate from the traditional classrooms to virtual learning spaces [11]. Indeed, the same situation obtained in most universities world over. In some universities, computer science departments have since then resorted to offering some of their courses online or on blended mode going forward. One such such course is CS1. This means that educators must now more than ever employ innovative methodologies to motivate students. Some teaching and learning methodologies that have gained popularity include video based learning and video game-based learning [85].

Never the less, *"limited research has been done to evaluate the impact of educational video games authored by teachers through instructor authoring tools on student perception and performance "* [85, p. 2]. In particular, no study has *"compared the effectiveness of video - based learning and game-based learning using educational video games authored by teachers through authoring tools "* [85, p. 2]. Further studies are necessary that could examine the use of game-based learning in online learning spaces. Using a quasi- experimental design with control and experimental groups, two possible questions could be investigated in future research [85], namely:

Question 1: *"Is game-based learning using teacher authored video games more effec-*

tive than video-based learning in terms of knowledge acquisition for CS1 students in online settings"?

Question 2: "*Is game based learning using teacher-authored video games more effective than video-game based learning in terms of motivation for CS1 students in online settings*"?

iv) In-person longitudinal studies

In future, rigorous in-person longitudinal studies could be conducted with students to see whether the generated games may be effective for a particular group of students [108]. Such studies could examine important aspects (student motivation, attitudes, knowledge and skills, satisfaction and perception [108]) change when using the generated games over a period of time. In this regard, controlled experiments could be done in future to investigate the impact on students learning gain when teachers create programming games using the tool and students learn with the generated games. Such experiments were not possible in the current thesis due to the limited time and the confounding variables which needed careful control to truly observe the effect of the games on student learning achievement. These type of studies need time and could possibly span two or more semesters [147].

v)Future Development

Based on the results of this thesis, one possible avenue of future research is a production quality game generator tool. This could involve making enhancements to some features of the prototype generator tool, especially the parts that were not completely designed due to time constraints. For example, making improvements to allow educators to easily create serious games from scratch. The work reported in this thesis mainly tested game generation by teachers when using given game examples. It would be worthwhile to enable CS teachers to generate their own games from scratch by specifying game assets and the level: (i) programming concept the students will learn, (ii) description, (iii) programming task (iii) difficulty, and (iv) programming task auto-marker. This would further give teachers more flexibility and autonomy to design personalised pedagogy given that they are the domain experts [148, 164]. In addition, teachers would be able to generate games for additional programming topics other than recursion. Regarding the current game examples that teachers can use to create their own games, more examples and levels could also be designed as an improvement. Finally, regarding the generated games, sound effects could be included to improve the student game play experience.

7.5.2 Reflection

This thesis presented findings from the design and evaluation of a novel prototype game generator tool called RGG to support programming instructors in higher education. The dissertation addresses an area of genuine interest within the CSE community - the use of GBL approach in teaching. Teaching and learning programming ought to take place in a fun and an engaging manner. To realise this goal, CS educators could take advantage of GBL. However, the adoption of GBL in mainstream teaching still remains low. Teachers who may wish to use GBL approaches are limited by their game programming skills and the lack of game authoring tools to support them. We reflect that having a game generator could effectively enable educators to easily create customised serious games.

The prototype in this thesis is designed to allow teachers to create and download the games. This is significant in the context of middle-income and poor countries given the lack of internet access [191]. In addition, this solves the adoption problems of delivery, distribution and classroom implementation [261]. This could be generalised for universal solutions. Moreover, as a teaching philosophy, the author believes that all learners wherever they are in the world should have access to hands-on learning materials. This is particularly the case if students from poor and middle-income countries are to succeed in learning programming with games as their counterparts in developed parts of the world. In view of completely packaged versus completely customised games, RGG produces completely customised games. As such, the tool can enable teachers to easily create instances of games by specifying game attributes. Such attributes include: game scenario, theme, background, color, genre, character, complexity of levels, among others. Customisation makes it easy for educators with limited game programming skills to easily configure various game aspects [3]. In addition, complete customisation would give teachers the opportunity to adapt the learning materials according to their learners needs. Whereas packaged tools will appeal to the needs of the general public, customised ones are tailored to the needs of a user group. Consequently, customised game authoring tools would be preferable as long as they cater to the needs of both novice and specialised users [111, 181].

Overall, despite the generalisation limitations, the evidence provided in this thesis is valid and useful for advancing GBL in the context of Africa and could be tested globally by the CSE research community. It is hoped that the insights drawn will ignite scholarly debate within the community.

REFERENCES

[1] A Gunter, G., D Robert, P., Kenny, F., D Erik, P., and Vick, H. A case for a formal design paradigm for serious games. 09 2018.

[2] Aedo Lopez, M., Vidal Duarte, E., Castro Gutierrez, E., and Paz Valderrama, A. Teaching abstraction, function and reuse in the first class of cs1: A lightbot experience. In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*, pages 256–257, 2016.

[3] Ahmad, A. and Law, E. L.-C. Educators as gamemasters: Creating serious role playing game with. *Proceedings of the ACM on Human-Computer Interaction*, 5 (CHI PLAY):1–29, 2021.

[4] Ahmad, A., Law, E. L., and Moseley, A. Integrating instructional design principles in serious games authoring tools: Insights from systematic literature review. In *Proceedings of the 11th Nordic Conference on Human-Computer Interaction: Shaping Experiences, Shaping Society*, pages 1–12, 2020.

[5] Ainsworth, S. and Fleming, P. Evaluating authoring tools for teachers as instructional designers. *Computers in human behavior*, 22(1):131–148, 2006.

[6] Ainsworth, S., Major, N., Grimshaw, S., Hayes, M., Underwood, J., Williams, B., and Wood, D. Redeem: Simple intelligent tutoring systems from usable tools. In *Authoring Tools for Advanced Technology Learning Environments*, pages 205–232. Springer, 2003.

[7] Albayrak, O. Instructor's acceptance of games utilization in undergraduate software engineering education: A pilot study in turkey. In *Proceedings of the Fourth International Workshop on Games and Software Engineering*, GAS '15, page 43–49. IEEE Press, 2015.

[8] Albert, W. and Tullis, T. *Measuring the user experience: collecting, analyzing, and presenting usability metrics*. Newnes, 2013.

[9] Allen, J. M., Vahid, F., Edgcomb, A., Downey, K., and Miller, K. An analysis of using many small programs in cs1. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, SIGCSE '19, page 585–591,

New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450358903. doi: 10.1145/3287324.3287466. URL https://doi.org/10.1145/3287324.3287466.

[10] Ambrosio, A. P., Costa, F. M., Almeida, L., Franco, A., and Macedo, J. Identifying cognitive abilities to improve cs1 outcome. In *2011 Frontiers in Education Conference (FIE)*, pages F3G–1–F3G–7, Oct 2011. doi: 10.1109/FIE.2011.6142824.

[11] Amimo, C. A. From the classroom into virtual learning environments: Essential knowledge, competences, skills and pedagogical strategies for the 21st century teacher education in kenya. In *Teacher Education in the 21st Century-Emerging Skills for a Changing World*. IntechOpen, 2021.

[12] Amineh, R. J. and Asl, H. D. Review of constructivism and social constructivism. *Journal of Social Sciences, Literature and Languages*, 1(1):9–16, 2015.

[13] Anderson, B. R., Karzmark, C. R., and Wardrip-Fruin, N. The psychological reality of procedural rhetoric. In *Proceedings of the 14th International Conference on the Foundations of Digital Games*, FDG '19, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450372176. doi: 10.1145/3337722.3337751. URL https://doi.org/10.1145/3337722.3337751.

[14] Anderson, J. R., Corbett, A. T., Koedinger, K. R., and Pelletier, R. Cognitive tutors: Lessons learned. *The journal of the learning sciences*, 4(2):167–207, 1995.

[15] Anderson, R. Thematic content analysis (tca). *Descriptive presentation of qualitative data*, 2007.

[16] Anyango, J. and Suleman, H. Supporting cs trainee teachers with game authoring tools. 2021.

[17] Anyango, J. T. and Suleman, H. Teaching programming in kenya and south africa: What is difficult and is it universal? In *Proceedings of the 18th Koli Calling International Conference on Computing Education Research*, pages 1–2, 2018.

[18] Anyango, J. T. and Suleman, H. Designing programming games for diversity in teaching introductory programming. 2020.

[19] Anyango, J. T. and Suleman, H. Supporting cs1 instructors: Design and evaluation of a game generator. In *Proceedings of the 26th ACM Conference on*

Innovation and Technology in Computer Science Education V. 1, pages 115–121, 2021.

[20] Arnold, M. L. et al. Measuring usability with the use questionnaire. *Tersedia di https://www. researchgate. net/profile/Arnold_Lund/publication/230786746_Measuring_Usability_with_the_USE_Questionnaire. Diakses*, 20, 2001.

[21] Baek, E.-O., Cagiltay, K., Boling, E., and Frick, T. User-centered design and development. *Handbook of research on educational communications and technology*, 1:660–668, 2008.

[22] Baek, Y. K. What hinders teachers in using computer and video games in the classroom? exploring factors inhibiting the uptake of computer and video games. *CyberPsychology and Behavior*, 11(6):665–671, 2008.

[23] Bakar, A., Inal, Y., and Cagiltay, K. Use of commercial games for educational purposes: Will today's teacher candidates use them in the future? In *EdMedia+ Innovate Learning*, pages 1757–1762. Association for the Advancement of Computing in Education (AACE), 2006.

[24] Baldeón, J., Puig, A., Rodríguez, I., Muriel, C., and Zardain, L. A conceptual model for educational game authoring: A showcase in math games. In Marcus, A. and Wang, W., editors, *Design, User Experience, and Usability: Designing Pleasurable Experiences*, pages 347–361, Cham, 2017. Springer International Publishing. ISBN 978-3-319-58637-3.

[25] Ball, A. F. Preparing teachers for diversity: Lessons learned from the us and south africa. *Teaching and Teacher Education*, 16(4):491–509, 2000.

[26] Barnes, T., Richter, H., Powell, E., Chaffin, A., and Godwin, A. Game2learn: Building cs1 learning games for retention. In *Proceedings of the 12th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, ITiCSE '07, pages 121–125, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-610-3. doi: 10.1145/1268784.1268821. URL `http://doi.acm.org/10.1145/1268784.1268821`.

[27] Barnes, T., Powell, E., Chaffin, A., and Lipford, H. Game2learn: Improving the motivation of cs1 students. In *Proceedings of the 3rd International Conference on Game Development in Computer Science Education*, GDCSE '08, pages 1–5, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-057-9. doi: 10.1145/1463673.1463674. URL `http://doi.acm.org/10.1145/1463673.1463674`.

[28] Barr, V. and Stephenson, C. Bringing computational thinking to k-12: What is involved and what is the role of the computer science education community? *ACM Inroads*, 2(1):48–54, Feb. 2011. ISSN 2153-2184. doi: 10.1145/1929887.1929905. URL http://doi.acm.org/10.1145/1929887.1929905.

[29] Battistella, P. and von Wangenheim, C. G. Games for teaching computing in higher education–a systematic review. *IEEE Technology and Engineering Education Journal*, 9(1):8–30, 2016.

[30] Bayliss, J. D. Using games in introductory courses: tips from the trenches. In *Proceedings of the 40th ACM technical symposium on Computer science education*, pages 337–341, 2009.

[31] Bayliss, J. D. and Strout, S. Games as a" flavor" of cs1. In *Proceedings of the 37th SIGCSE technical symposium on Computer science education*, pages 500–504, 2006.

[32] Beaubouef, T. and Mason, J. Why the high attrition rate for computer science students: some thoughts and observations. *ACM SIGCSE Bulletin*, 37(2):103–106, 2005.

[33] Becker, B. A. A survey of introductory programming courses in ireland. In *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*, ITiCSE '19, page 58–64, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450368957. doi: 10.1145/3304221.3319752. URL https://doi.org/10.1145/3304221.3319752.

[34] Bennedsen, J. and Caspersen, M. E. Failure rates in introductory programming. *AcM SIGcSE Bulletin*, 39(2):32–36, 2007.

[35] Bennedsen, J. and Caspersen, M. E. Failure rates in introductory programming: 12 years later. *ACM Inroads*, 10(2):30–36, apr 2019. ISSN 2153-2184. doi: 10.1145/3324888. URL https://doi.org/10.1145/3324888.

[36] Bevan, N. Classifying and selecting ux and usability measures. In *International Workshop on Meaningful Measures: Valid Useful User Experience Measurement*, volume 11, pages 13–18, 2008.

[37] Bontchev, B. and Panayotova, R. Generation of educational 3d maze games for carpet handicraft in bulgaria. *Digital Presentation and Preservation of Cultural and Scientific Heritage*, (7):41–52, 2017.

[38] Bonwell, C. C. and Eison, J. A. *Active Learning: Creating Excitement in the Classroom. 1991 ASHE-ERIC Higher Education Reports.* ERIC, 1991.

[39] Bosse, Y., Redmiles, D., and Gerosa, M. A. Pedagogical content for professors of introductory programming courses. In *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*, ITiCSE '19, pages 429–435, New York, NY, USA, 2019. ACM. ISBN 978-1-4503-6895-7. doi: 10.1145/3304221.3319776. URL `http://doi.acm.org/10.1145/3304221.3319776`.

[40] Bourgonjon, J., De Grove, F., De Smet, C., Van Looy, J., Soetaert, R., and Valcke, M. Acceptance of game-based learning by secondary school teachers. *Computers and education*, 67:21–35, 2013.

[41] Bouzid, Y., Khenissi, M. A., and Jemni, M. Designing a game generator as an educational technology for the deaf learners. In *Information and Communication Technology and Accessibility (ICTA), 2015 5th International Conference on*, pages 1–6. IEEE, 2015.

[42] Brown, E. and Cairns, P. A grounded investigation of game immersion. In *CHI'04 extended abstracts on Human factors in computing systems*, pages 1297–1300, 2004.

[43] Bruckman, A., Biggers, M., Ericson, B., McKlin, T., Dimond, J., DiSalvo, B., Hewner, M., Ni, L., and Yardi, S. " georgia computes!" improving the computing education pipeline. *ACM SIGCSE Bulletin*, 41(1):86–90, 2009.

[44] Brusilovsky, P., Grady, J., Spring, M., and Lee, C.-H. What should be visualized? faculty perception of priority topics for program visualization. *SIGCSE Bull.*, 38 (2):44–48, June 2006. ISSN 0097-8418. doi: 10.1145/1138403.1138431. URL `https://doi.org/10.1145/1138403.1138431`.

[45] Buffardi, K. and Chavan, S. Is programming relevant to cs1 students' interests? *Journal of Computing Sciences in Colleges*, 37(1):45–53, 2021.

[46] Camp, T. The incredible shrinking pipeline. *Communications of the ACM*, 40 (10):103–110, 1997.

[47] Carvalho, M. B., Bellotti, F., Berta, R., De Gloria, A., Sedano, C. I., Hauge, J. B., Hu, J., and Rauterberg, M. An activity theory-based model for serious games analysis and conceptual design. *Computers and education*, 87:166–181, 2015.

[48] Chaffin, A., Doran, K., Hicks, D., and Barnes, T. Experimental evaluation of teaching recursion in a video game. In *In Proceedings of the 2009 ACM SIGGRAPH Symposium on Video Games, Sandbox '09*, pages 79–86. ACM, 2009.

[49] Chaudy, Y., Connolly, T., and Hainey, T. An assessment engine: Educators as editors of their serious games' assessment. In *ECGBL2014-8th European Conference on Games Based Learning: ECGBL2014*, page 58. Academic Conferences and Publishing International, 2014.

[50] Cheah, C. S. Factors contributing to the difficulties in teaching and learning of computer programming: A literature review. *Contemporary Educational Technology*, 12(2):ep272, 2020.

[51] Chetty, J. and van der Westhuizen, D. Towards a pedagogical design for teaching novice programmers: Design-based research as an empirical determinant for success. In *Proceedings of the 15th Koli Calling Conference on Computing Education Research*, Koli Calling '15, pages 5–12, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-4020-5. doi: 10.1145/2828959.2828976. URL http://doi.acm.org/10.1145/2828959.2828976.

[52] Chorianopoulos, K., Giannakos, M. N., and Chrisochoides, N. Design principles for serious games in mathematics. In *Proceedings of the 18th Panhellenic Conference on Informatics*, PCI '14, page 1–5, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450328975. doi: 10.1145/2645791.2645843. URL https://doi.org/10.1145/2645791.2645843.

[53] Cohoon, J. P., Cohoon, J. M., and Soffa, M. L. Educating diverse computing students at the university of virginia. *Computer*, 46(3):52–55, 2013.

[54] Collain, M. and Trytten, D. "you don't have to be a white male that was learning how to program since he was five". In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, SIGCSE '19, pages 968–974, New York, NY, USA, 2019. ACM. ISBN 978-1-4503-5890-3. doi: 10.1145/3287324.3287383. URL http://doi.acm.org/10.1145/3287324.3287383.

[55] Cook, M., Colton, S., and Gow, J. The angelina videogame design system—part ii. *IEEE Transactions on Computational Intelligence and AI in Games*, 9(3):254–266, Sep. 2017. ISSN 1943-0698. doi: 10.1109/TCIAIG.2016.2520305.

[56] Cooper, S., Dann, W., and Pausch, R. Alice: a 3-d tool for introductory programming concepts. *Journal of computing sciences in colleges*, 15(5):107–116, 2000.

[57] Dale, N. B. Most difficult topics in cs1: Results of an online survey of educators. *SIGCSE Bull.*, 38(2):49–53, June 2006. ISSN 0097-8418. doi: 10.1145/1138403.1138432. URL https://doi.org/10.1145/1138403.1138432.

[58] Dann, W., Cooper, S., and Pausch, R. Using visualization to teach novices recursion. In *Proceedings of the 6th Annual Conference on Innovation and Technology in Computer Science Education*, ITiCSE '01, pages 109–112, New York, NY, USA, 2001. ACM. ISBN 1-58113-330-8. doi: 10.1145/377435.377507. URL http://doi.acm.org/10.1145/377435.377507.

[59] Davis, F. D. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS quarterly*, pages 319–340, 1989.

[60] De Freitas, S. and Oliver, M. How can exploratory learning with games and simulations within the curriculum be most effectively evaluated? *Computers and education*, 46(3):249–264, 2006.

[61] De Gloria, A., Bellotti, F., and Berta, R. Serious games for education and training. *International Journal of Serious Games*, 1(1), 2014.

[62] Dempsey, J. V., Lucassen, B., and Rasmussen, K. *The instructional gaming literature: Implications and 99 sources*. University of South Carolina, College of Education South Carolina, 1996.

[63] Denny, P., Becker, B. A., Bosch, N., Prather, J., Reeves, B., and Whalley, J. Novice reflections during the transition to a new programming language. SIGCSE 2022, page 948–954, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450390705. doi: 10.1145/3478431.3499314. URL https://doi.org/10.1145/3478431.3499314.

[64] Diefenbach, S., Kolb, N., and Hassenzahl, M. The 'hedonic' in human-computer interaction: History, contributions, and future research directions. In *Proceedings of the 2014 Conference on Designing Interactive Systems*, DIS '14, page 305–314, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450329026. doi: 10.1145/2598510.2598549. URL https://doi.org/10.1145/2598510.2598549.

[65] Djaouti, D., Alvarez, J., and Jessel, J.-P. Can "gaming 2.0" help design "serious games"? a comparative study. In *Proceedings of the 5th ACM SIGGRAPH Symposium on Video Games*, Sandbox '10, page 11–18, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781450300971. doi: 10.1145/1836135.1836137. URL https://doi.org/10.1145/1836135.1836137.

[66] Djelil, F. and Sanchez, E. Game design and didactic transposition of knowledge. the case of progo, a game dedicated to learning object-oriented programming. *Education and Information Technologies*, pages 1–20, 2022.

[67] Dlodlo, N. Access to ict education for girls and women in rural south africa: A case study. *Technology in society*, 31(2):168–175, 2009.

[68] Dondlinger, M. J. Educational video game design: A review of the literature. *Journal of applied educational technology*, 4(1):21–31, 2007.

[69] Dong, Y. and Barnes, T. Evaluation of a template-based puzzle generator for an educational programming game. In *Proceedings of the 12th International Conference on the Foundations of Digital Games*, page 40. ACM, 2017.

[70] Dos Passos Canteri, R., García, L. S., de Felipe, T. A., Galvao, L. F. O., and Antunes, D. R. Conceptual framework to support a web authoring tool of educational games for deaf children. In *CSEDU (2)*, pages 226–235, 2019.

[71] Döweling, S., Schmidt, B., and Göb, A. A model for the design of interactive systems based on activity theory. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, pages 539–548, 2012.

[72] Eaachak, L., Belhabibe, A., and Bouhorma, M. Towards a new concept of serious games generator. In *Electrical and Information Technologies (ICEIT), 2015 International Conference on*, pages 341–346. IEEE, 2015.

[73] Eagle, M. and Barnes, T. Experimental evaluation of an educational game for improved learning in introductory computing. In *Proceedings of the 40th ACM Technical Symposium on Computer Science Education*, SIGCSE '09, page 321–325, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605581835. doi: 10.1145/1508865.1508980. URL `https://doi.org/10.1145/1508865.1508980`.

[74] Fernández-Vara, C. and Thomson, A. Procedural generation of narrative puzzles in adventure games: The puzzle-dice system. In *Proceedings of the The third workshop on Procedural Content Generation in Games*, page 12. ACM, 2012.

[75] Fosnot, C. T. *Constructivism: Theory, perspectives, and practice*. Teachers College Press, 2013.

[76] Gaeta, M., Loia, V., Mangione, G. R., Orciuoli, F., Ritrovato, P., and Salerno, S. A methodology and an authoring tool for creating complex learning objects to support interactive storytelling. *Computers in Human Behavior*, 31:620–637, 2014.

[77] Gagne, R. The conditions of learning and theory of instruction robert gagné. *New York, NY: Holt, Rinehart ja Winston*, 1985.

[78] Galpin, V. Women in computing around the world. *ACM SIGCSE Bulletin*, 34 (2):94–100, 2002.

[79] Games, A. and Kane, L. Exploring adolescent's stem learning through scaffolded game design. In *Proceedings of the 6th International Conference on Foundations of Digital Games*, FDG '11, pages 1–8, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0804-5. doi: 10.1145/2159365.2159366. URL http://doi.acm.org/10.1145/2159365.2159366.

[80] Garcia-Gathright, J., Hosey, C., Thomas, B. S., Carterette, B., and Diaz, F. Mixed methods for evaluating user satisfaction. RecSys '18, page 541–542, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450359016. doi: 10.1145/3240323.3241622. URL https://doi.org/10.1145/3240323.3241622.

[81] Gee, J. P. What video games have to teach us about learning and literacy. *Comput. Entertain.*, 1(1):20–20, Oct. 2003. ISSN 1544-3574. doi: 10.1145/950566.950595. URL http://doi.acm.org/10.1145/950566.950595.

[82] Gibson, B. and Bell, T. Evaluation of games for teaching computer science. In *Proceedings of the 8th Workshop in Primary and Secondary Computing Education*, WiPSE '13, page 51–60, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450324557. doi: 10.1145/2532748.2532751. URL https://doi.org/10.1145/2532748.2532751.

[83] Gilleade, K. M. and Dix, A. Using frustration in the design of adaptive videogames. In *Proceedings of the 2004 ACM SIGCHI International Conference on Advances in computer entertainment technology*, pages 228–232, 2004.

[84] Ginat, D. and Shifroni, E. Teaching recursion in a procedural environmentandm-dash;how much should we emphasize the computing model? *SIGCSE Bull.*, 31 (1):127–131, Mar. 1999. ISSN 0097-8418. doi: 10.1145/384266.299718. URL http://doi.acm.org/10.1145/384266.299718.

[85] Gordillo, A., López-Fernández, D., and Tovar, E. Comparing the effectiveness of video-based learning and game-based learning using teacher-authored video games for online software engineering education. *IEEE Transactions on Education*, 2022.

[86] Gouws, L. A., Bradshaw, K., and Wentworth, P. Computational thinking in educational activities: an evaluation of the educational game light-bot. In *Proceedings of the 18th ACM conference on Innovation and technology in computer science education*, pages 10–15, 2013.

[87] Green, M. C., Khalifa, A., Barros, G. A., Nealen, A., and Togelius, J. Generating levels that teach mechanics. In *Proceedings of the 13th International Conference on the Foundations of Digital Games*, page 55. ACM, 2018.

[88] Gunter, G. A., Kenny, R. F., and Vick, E. H. Taking educational games seriously: using the retain model to design endogenous fantasy into standalone educational games. *Educational Technology Research and Development*, 56(5): 511–537, Dec 2008. ISSN 1556-6501. doi: 10.1007/s11423-007-9073-2. URL `https://doi.org/10.1007/s11423-007-9073-2`.

[89] Guzdial, M. and Soloway, E. Teaching the nintendo generation to program. *Communications of the ACM*, 45(4):17–21, 2002.

[90] Hadjerrouit, S. A constructivist framework for integrating the java paradigm into the undergraduate curriculum. In *Proceedings of the 6th Annual Conference on the Teaching of Computing and the 3rd Annual Conference on Integrating Technology into Computer Science Education: Changing the Delivery of Computer Science Education*, ITiCSE '98, pages 105–107, New York, NY, USA, 1998. ACM. ISBN 1-58113-000-7. doi: 10.1145/282991.283079. URL `http://doi.acm.org/10.1145/282991.283079`.

[91] Halff, H. M., Hsieh, P. Y., Wenzel, B. M., Chudanov, T. J., Dirnberger, M. T., Gibson, E. G., and Redfield, C. L. Requiem for a development system: reflections on knowledge-based, generative instruction. In *Authoring tools for advanced technology learning environments*, pages 33–59. Springer, 2003.

[92] Hamari, J., Shernoff, D. J., Rowe, E., Coller, B., Asbell-Clarke, J., and Edwards, T. Challenging games help students learn: An empirical study on engagement, flow and immersion in game-based learning. *Computers in human behavior*, 54: 170–179, 2016.

[93] Harteveld, C., Smith, G., Carmichael, G., Gee, E., and Stewart-Gardiner, C. A design-focused analysis of games teaching computer science. *Proceedings of Games+ Learning+ Society*, 10, 2014.

[94] Hassenzahl, M. The interplay of beauty, goodness, and usability in interactive products. *Human–Computer Interaction*, 19(4):319–349, 2004.

[95] Hosseini, H. and Perweiler, L. Are you game? In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, pages 866–872, 2019.

[96] Hotte, R., Ferreira, S. M., Abdessettar, S., and Gouin-Vallerand, C. Digital learning game scenario: A pedagogical pattern applied to serious game design. 2017.

[97] Hovey, C. L., Barker, L., and Luebs, M. Frequency of instructor-and student-centered teaching practices in introductory cs courses. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, pages 599–605, 2019.

[98] Hovey, C. L., Barker, L., and Nagy, V. Survey results on why cs faculty adopt new teaching practices. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, pages 483–489, 2019.

[99] Huitt, W. Bloom et al.'s taxonomy of the cognitive domain. *Educational psychology interactive*, 22, 2011.

[100] Hussain, J., Khan, W. A., Hur, T., Bilal, H. S. M., Bang, J., Hassan, A. U., Afzal, M., and Lee, S. A multimodal deep log-based user experience (ux) platform for ux evaluation. *Sensors*, 18(5):1622, 2018.

[101] Ibe, N. A., Howsmon, R., Penney, L., Granor, N., DeLyser, L. A., and Wang, K. Reflections of a diversity, equity, and inclusion working group based on data from a national cs education program. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, SIGCSE '18, pages 711–716, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-5103-4. doi: 10.1145/3159450.3159594. URL `http://doi.acm.org/10.1145/3159450.3159594`.

[102] Ibrahim, R. and Jaafar, A. Educational games (eg) design framework: Combination of game design, pedagogy and content modeling. In *2009 International Conference on Electrical Engineering and Informatics*, volume 1, pages 293–298. IEEE, 2009.

[103] IJsselsteijn, W. A., de Kort, Y. A., and Poels, K. The game experience questionnaire. *Eindhoven: Technische Universiteit Eindhoven*, 46(1), 2013.

[104] Inan Nur, A., B. Santoso, H., and O. Hadi Putra, P. The method and metric of user experience evaluation: A systematic literature review. In *2021 10th International Conference on Software and Computer Applications*, ICSCA 2021, page 307–317, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450388825. doi: 10.1145/3457784.3457832. URL `https://doi.org/10.1145/3457784.3457832`.

[105] Iqbal, S. and Coldwell-Neilson, J. A model for teaching an introductory programming course using adri. 22, 02 2016.

[106] Isbister, K., Flanagan, M., and Hash, C. Designing games for learning: Insights from conversations with designers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, page 2041–2044, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781605589299. doi: 10.1145/1753326.1753637. URL https://doi.org/10.1145/1753326.1753637.

[107] Jenkins, T. On the difficulty of learning to program. In *Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences*, volume 4, pages 53–58. Citeseer, 2002.

[108] Johnson, C., McGill, M., Bouchard, D., Bradshaw, M. K., Bucheli, V. A., Merkle, L. D., Scott, M. J., Sweedyk, Z., Ángel, J., Xiao, Z., et al. Game development for computer science education. In *Proceedings of the 2016 ITiCSE Working Group Reports*, pages 23–44. ACM, 2016.

[109] Kaimara, P., Fokides, E., Oikonomou, A., and Deliyannis, I. Pre-service teachers' views about the use of digital educational games for collaborative learning. *Education and Information Technologies*, 27(4):5397–5416, 2022.

[110] Karoui, A., Marfisi-Schottman, I., and George, S. Mobile learning game authoring tools: assessment, synthesis and proposals. In *International Conference on Games and Learning Alliance*, pages 281–291. Springer, 2016.

[111] Karoui, A., Marfisi-Schottman, I., and George, S. A nested design approach for mobile learning games. In *Proceedings of the 16th World Conference on Mobile and Contextual Learning*, mLearn 2017, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450352550. doi: 10.1145/3136907.3136923. URL https://doi.org/10.1145/3136907.3136923.

[112] Katz, J. *Teaching to diversity: The three-block model of universal design for learning*. Portage and Main Press, 2012.

[113] Kazimoglu, C., Kiernan, M., Bacon, L., and Mackinnon, L. A serious game for developing computational thinking and learning introductory computer programming. *Procedia-Social and Behavioural Sciences*, 47:1991–1999, 2012.

[114] Keller, J. M. Development and use of the arcs model of instructional design. *Journal of instructional development*, 10(3):2–10, 1987.

[115] Keller, J. M. Using the arcs process in cbi and distance education. *Motivation in teaching and learning: New directions for teaching and learning. San Francisco: Jossey-Bass*, 536, 1998.

[116] Khalifa, A., Perez-Liebana, D., Lucas, S. M., and Togelius, J. General video game level generation. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, pages 253–259. ACM, 2016.

[117] Khenissi, M. A., Essalmi, F., and Jemni, M. Comparison between serious games and learning version of existing games. *Procedia-Social and Behavioral Sciences*, 191:487–494, 2015.

[118] Kim, A. S. and Ko, A. J. A pedagogical analysis of online coding tutorials. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, pages 321–326, 2017.

[119] Kinnunen, P. and Malmi, L. Why students drop out cs1 course? ICER '06, page 97–108, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595934944. doi: 10.1145/1151588.1151604. URL https://doi.org/10.1145/1151588.1151604.

[120] Kletenik, D. and Sturm, D. Game development with a serious focus. SIGCSE '18, page 652–657, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450351034. doi: 10.1145/3159450.3159588. URL https://doi.org/10.1145/3159450.3159588.

[121] Kolb, A. Y. and Kolb, D. A. Learning styles and learning spaces: Enhancing experiential learning in higher education. *Academy of management learning and education*, 4(2):193–212, 2005.

[122] Kortum P, G. M. and F, O. Psychometric evaluation of the use (usefulness, satisfaction, and ease of use) questionnaire for reliability and validity. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, page 1414–1418, 2018.

[123] Koulouri, T., Lauria, S., and Macredie, R. D. Teaching introductory programming: A quantitative evaluation of different approaches. *ACM Transactions on Computing Education (TOCE)*, 14(4):1–28, 2014.

[124] Lahtinen, E., Ala-Mutka, K., and Järvinen, H.-M. A study of the difficulties of novice programmers. *SIGCSE Bull.*, 37(3):14–18, June 2005. ISSN 0097-8418. doi: 10.1145/1151954.1067453. URL http://doi.acm.org/10.1145/1151954.1067453.

[125] Lallemand, C., Gronier, G., and Koenig, V. User experience: A concept without consensus? exploring practitioners' perspectives through an international survey. *Computers in Human Behavior*, 43:35–48, 2015.

[126] Lameras, P., Arnab, S., Dunwell, I., Stewart, C., Clarke, S., and Petridis, P. Essential features of serious games design in higher education: Linking learning attributes to game mechanics. *British Journal of Educational Technology*, 48(4): 972–994, 2017. doi: 10.1111/bjet.12467. URL `https://onlinelibrary.wiley.com/doi/abs/10.1111/bjet.12467`.

[127] Landro, A. C. S. A game generator-the framework for an educational system development game. Master's thesis, 2007.

[128] Laporte, L. and Zaman, B. Informing content-driven design of computer programming games: a problems analysis and a game review. In *Proceedings of the 9th Nordic Conference on Human-Computer Interaction*, page 61. ACM, 2016.

[129] Lawan, A. A., Abdi, A. S., Abuhassan, A. A., and Khalid, M. S. What is difficult in learning programming language based on problem-solving skills? In *2019 International Conference on Advanced Science and Engineering (ICOASE)*, pages 18–22. IEEE, 2019.

[130] Lee, E., Shan, V., Beth, B., and Lin, C. A structured approach to teaching recursion using cargo-bot. In *Proceedings of the Tenth Annual Conference on International Computing Education Research*, ICER '14, pages 59–66, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2755-8. doi: 10.1145/2632320.2632356. URL `http://doi.acm.org/10.1145/2632320.2632356`.

[131] Leping, V., Lepp, M., Niitsoo, M., Tõnisson, E., Vene, V., and Villems, A. Python prevails. In *Proceedings of the International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing*, pages 1–5, 2009.

[132] Lepp, M. and Kaimre, J. Providing additional support in an introductory programming course. In *2022 IEEE Global Engineering Education Conference (EDUCON)*, pages 210–216. IEEE, 2022.

[133] Lonati, V., Malchiodi, D., Monga, M., and Morpurgo, A. Bebras as a teaching resource: classifying the tasks corpus using computational thinking skills. In *Proceedings of the 22nd annual conference on innovation and technology in computer science education (ITiCSE 2017)*, page 366, 2017. ISBN 978-1-4503-4704-4. doi: http://dx.doi.org/10.1145/3059009.3072987.

[134] López-Fernández, D., Gordillo, A., Alarcón, P. P., and Tovar, E. Comparing traditional teaching and game-based learning using teacher-authored games on computer science education. *IEEE Transactions on Education*, 2021.

[135] Lund, A. M. Measuring usability with the use questionnaire12. *Usability interface*, 8(2):3–6, 2001.

[136] Luo, J., Bellows, L., and Grady, M. Classroom management issues for teaching assistants. *Research in Higher Education*, 41(3):353–383, 2000.

[137] Luxton-Reilly, A. Learning to program is easy. In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*, pages 284–289, 2016.

[138] Luxton-Reilly, A. and Adelakun-Adeyemo, O. Confirmation bias and other flaws in citing pass rate studies. In *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1*, pages 575–581, 2021.

[139] Luxton-Reilly, A., Albluwi, I., Becker, B. A., Giannakos, M., Kumar, A. N., Ott, L., Paterson, J., Scott, M. J., Sheard, J., and Szabo, C. Introductory programming: a systematic literature review. In *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*, pages 55–106, 2018.

[140] Lyulina, E., Birillo, A., Kovalenko, V., and Bryksin, T. Tasktracker-tool: A toolkit for tracking of code snapshots and activity data during solution of programming tasks. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, SIGCSE '21, page 495–501, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450380621. doi: 10.1145/3408877.3432534. URL `https://doi.org/10.1145/3408877.3432534`.

[141] MacDonald, C. M. and Atwood, M. E. What does it mean for a system to be useful? an exploratory study of usefulness. In *Proceedings of the 2014 conference on Designing interactive systems*, pages 885–894, 2014.

[142] Mackay, S. What does literature tell us about recursion? In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 2*, pages 1173–1173, 2022.

[143] Maguire, M. Context of use within usability activities. *International Journal of Human-Computer Studies*, 55(4):453–483, 2001.

[144] Maguire, M. Methods to support human-centred design. *International journal of human-computer studies*, 55(4):587–634, 2001.

[145] Maia, L. F., Nolêto, C., Lima, M., Ferreira, C., Marinho, C., Viana, W., and Trinta, F. Lagarto: A location based games authoring tool enhanced with augmented reality features. *Entertainment computing*, 22:3–13, 2017.

[146] Malik, S. I. and Coldwell-Neilson, J. A model for teaching an introductory programming course using adri. *Education and Information Technologies*, 22(3): 1089–1120, 2017.

[147] Malik, S. I., Mathew, R., Al-Sideiri, A., Jabbar, J., Al-Nuaimi, R., and Tawafak, R. M. Enhancing problem-solving skills of novice programmers in an introductory programming course. *Computer Applications in Engineering Education*, 30 (1):174–194, 2022.

[148] Malliarakis C., X. S., Satratzemi M. Educational games for teaching computer programming. In *Research on e-Learning and ICT in Education*, pages 87–98. Springer, 2014.

[149] Mao, J.-Y., Vredenburg, K., Smith, P. W., and Carey, T. The state of user-centered design practice. *Commun. ACM*, 48(3):105–109, Mar. 2005. ISSN 0001-0782. doi: 10.1145/1047671.1047677. URL `https://doi.org/10.1145/1047671.1047677`.

[150] Marchiori, E. J., Torrente, J., del Blanco, Á., Moreno-Ger, P., Sancho, P., and Fernández-Manjón, B. A narrative metaphor to facilitate educational game authoring. *Computers and Education*, 58(1):590–599, 2012.

[151] Marín-Vega, H., Alor-Hernández, G., Colombo-Mendoza, L. O., Sánchez-Ramírez, C., García-Alcaraz, J. L., and Avelar-Sosa, L. Zeus–a tool for generating rule-based serious games with gamification techniques. *IET Software*, 14 (2):88–97, 2019.

[152] Marín-Vega, H., Alor-Hernández, G., Colombo-Mendoza, L. O., Bustos-López, M., and Zataraín-Cabada, R. Zeusar: a process and an architecture to automate the development of augmented reality serious games. *Multimedia Tools and Applications*, 81(2):2901–2935, 2022.

[153] Mark, B., Berechet, T., Mahlmann, T., and Togelius, J. Procedural generation of 3d caves for games on the gpu. In *FDG*, 2015.

[154] Marti, P. and Iacono, I. Evaluating the experience of use of a squeezable interface. In *Proceedings of the 11th Biannual Conference on Italian SIGCHI Chapter*, pages 42–49, 2015.

[155] Mathew, R., Malik, S. I., and Tawafak, R. M. Teaching problem solving skills using an educational game in a computer programming course. *Informatics in education*, 18(2):359–373, 2019.

[156] Matthíasdóttir, A. and Geirsson, H. J. The novice problem in computer science. In *Proceedings of the 12th International Conference on Computer Systems and Technologies*, CompSysTech '11, pages 570–576, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0917-2. doi: 10.1145/2023607.2023702. URL http://doi.acm.org/10.1145/2023607.2023702.

[157] Mbogo, C., Blake, E., and Suleman, H. A mobile scaffolding application to support novice learners of computer programming. In *Proceedings of the Sixth International Conference on Information and Communications Technologies and Development: Notes - Volume 2*, ICTD '13, page 84–87, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450319072. doi: 10.1145/2517899.2517941. URL https://doi.org/10.1145/2517899.2517941.

[158] Mbogo, C. C. Scaffolding java programming on a mobile phone for novice learners. 2015.

[159] McCall, D. and Kölling, M. A new look at novice programmer errors. *ACM Trans. Comput. Educ.*, 19(4):38:1–38:30, July 2019. ISSN 1946-6226. doi: 10.1145/3335814. URL http://doi.acm.org/10.1145/3335814.

[160] McCauley, R., Grissom, S., Fitzgerald, S., and Murphy, L. Teaching and learning recursive programming: a review of the research literature. *Computer Science Education*, 25(1):37–66, 2015. doi: 10.1080/08993408.2015.1033205. URL https://doi.org/10.1080/08993408.2015.1033205.

[161] McCracken, D. D. Ruminations on computer science curricula. *Communications of the ACM*, 30(1):3–6, 1987.

[162] McGettrick, A., Boyle, R., Ibbett, R., Lloyd, J., Lovegrove, G., and Mander, K. Grand challenges in computing–education [grandes retos en la educación de la computación]. *UK: The British Computer Society*, 2004.

[163] McGill, M. M., Johnson, C., Atlas, J., Bouchard, D., Messom, C., Pollock, I., and Scott, M. J. If memory serves: Towards designing and evaluating a game for

teaching pointers to undergraduate students. In *Proceedings of the 2017 ITiCSE Conference on Working Group Reports*, ITiCSE-WGR '17, pages 25–46, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-5627-5. doi: 10.1145/3174781.3174783. URL http://doi.acm.org/10.1145/3174781.3174783.

[164] Medeiros, R. P., Ramalho, G. L., and Falcão, T. P. A systematic literature review on teaching and learning introductory programming in higher education. *IEEE Transactions on Education*, 62(2):77–90, 2019.

[165] Mehm, F. Authoring serious games. In *Proceedings of the Fifth International Conference on the Foundations of Digital Games*, FDG '10, page 271–273, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781605589374. doi: 10.1145/1822348.1822390. URL https://doi.org/10.1145/1822348.1822390.

[166] Mehm F., G. S. S. R., Reuter C. Future trends in game authoring tools. *Springer*, 7522:557–629, 2012.

[167] Melero, J., Hern'ndez-Leo, D., and Blat, J. Considerations for the design of mini-games integrating hints for puzzle solving ict-related concepts. In *Advanced Learning Technologies (ICALT), 2012 IEEE 12th International Conference on*, pages 154–158. IEEE, 2012.

[168] Menestrina, Z. and Angeli, A. D. End-user development for serious games. In *New Perspectives in End-User Development*, pages 359–383. Springer, 2017.

[169] Merrill, M. D. Using knowledge objects to design instructional learning environments. In *Authoring Tools for Advanced Technology Learning Environments*, pages 181–204. Springer, 2003.

[170] Mildner, P. and 'Floyd' Mueller, F. *Design of Serious Games*, pages 57–82. Springer International Publishing, Cham, 2016. ISBN 978-3-319-40612-1. doi: 10.1007/978-3-319-40612-1_3. URL https://doi.org/10.1007/978-3-319-40612-1_3.

[171] Miller, A., Reges, S., and Obourn, A. jgrasp: a simple, visual, intuitive programming environment for cs1 and cs2. *ACM Inroads*, 8(4):53–58, 2017.

[172] Miller, C. S., Settle, A., and Lalor, J. Learning object-oriented programming in python: Towards an inventory of difficulties and testing pitfalls. In *Proceedings of the 16th Annual Conference on Information Technology Education*, SIGITE '15, pages 59–64, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3835-6. doi: 10.1145/2808006.2808017. URL http://doi.acm.org/10.1145/2808006.2808017.

[173] Milne, I. and Rowe, G. Difficulties in learning and teaching programming—views of students and tutors. *Education and Information technologies*, 7(1):55–66, 2002.

[174] Mirel, B. *Interaction design for complex problem solving: Developing useful and usable software*. Morgan Kaufmann, 2004.

[175] Mirolo, C. Learning (through) recursion: A multidimensional analysis of the competences achieved by cs1 students. In *Proceedings of the Fifteenth Annual Conference on Innovation and Technology in Computer Science Education*, ITiCSE '10, pages 160–164, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-820-9. doi: 10.1145/1822090.1822136. URL http://doi.acm.org/10.1145/1822090.1822136.

[176] Mohamed, A. Designing a cs1 programming course for a mixed-ability class. In *Proceedings of the Western Canadian Conference on Computing Education*, WCCCE '19, pages 8:1–8:6, New York, NY, USA, 2019. ACM. ISBN 978-1-4503-6715-8. doi: 10.1145/3314994.3325084. URL http://doi.acm.org/10.1145/3314994.3325084.

[177] Molnar, A., Farrell, D., and Kostova, P. Who poisoned hugh?-the star framework: integrating learning objectives with storytelling. In *International Conference on Interactive Digital Storytelling*, pages 60–71. Springer, 2012.

[178] Monge, A. E., Fadjo, C. L., Quinn, B. A., and Barker, L. J. Engagecsedu: Engaging and retaining cs1 and cs2 students. *ACM Inroads*, 6(1):6–11, Feb. 2015. ISSN 2153-2184. doi: 10.1145/2714569. URL http://doi.acm.org/10.1145/2714569.

[179] Murray, T. Authoring intelligent tutoring systems: An analysis of the state of the art. *International Journal of Artificial Intelligence in Education (IJAIED)*, 10: 98–129, 1999.

[180] Murray, T. Design tradeoffs in usability and power for advanced educational software authoring tools. *Educational Technology*, 44(5):10–16, 2004.

[181] Murray, T. Coordinating the complexity of tools, tasks, and users: On theory-based approaches to authoring tool usability. *International Journal of Artificial Intelligence in Education*, 26(1):37–71, 2016.

[182] Mutanu, L. and Machoka, P. Enhancing computer students' academic performance through explanatory modeling. In Tait, B., Kroeze, J., and Gruner, S., editors, *ICT Education*, pages 227–243, Cham, 2020. Springer International Publishing. ISBN 978-3-030-35629-3.

[183] Nacke, L. and Lindley, C. Boredom, immersion, flow: A pilot study investigating player experience. In *IADIS International Conference Gaming 2008: Design for engaging experience and social interaction*. IADIS Press, 2008.

[184] Nacke, L. and Lindley, C. A. Flow and immersion in first-person shooters: measuring the player's gameplay experience. In *Proceedings of the 2008 conference on future play: Research, play, share*, pages 81–88, 2008.

[185] Nancarrow, C. and Brace, I. Saying the "right thing": Coping with social desirability bias in marketing research. *Bristol Business School Teaching and Research Review*, 3(11):1–11, 2000.

[186] Naps, T. L., Rößling, G., Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C., Korhonen, A., Malmi, L., McNally, M., Rodger, S., et al. Exploring the role of visualization and engagement in computer science education. In *Working group reports from ITiCSE on Innovation and technology in computer science education*, pages 131–152. 2002.

[187] National Science Foundation. Women, minorities, and persons with disabilities in science and engineering: 2013.technical report, national center for science and engineering statistics, directorate for social, behavioral and economics science, 2013. Technical report, 2013.

[188] Newhall, T., Meeden, L., Danner, A., Soni, A., Ruiz, F., and Wicentowski, R. A support program for introductory cs courses that improves student performance and retains students from underrepresented groups. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, SIGCSE '14, pages 433–438, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2605-6. doi: 10.1145/2538862.2538923. URL `http://doi.acm.org/10.1145/2538862.2538923`.

[189] Ni, L. What makes cs teachers change? factors influencing cs teachers' adoption of curriculum innovations. SIGCSE '09, page 544–548, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605581835. doi: 10.1145/1508865.1509051. URL `https://doi.org/10.1145/1508865.1509051`.

[190] Nielsen, J. *Usability engineering*. Morgan Kaufmann, 1994.

[191] Nomusa, D. Access to ict education for girls and women in rural south africa: A case study. *Elsevier*, 31(2):168–175, May 2009. doi: 10.1016/j.techsoc.2009.03.003.

[192] Novak, J. and Schmidt, S. When joy matters: The importance of hedonic stimulation in collocated collaboration with large-displays. In *IFIP Conference on Human-Computer Interaction*, pages 618–629. Springer, 2009.

[193] Oblinger, D. The next generation of educational engagement. *Journal of interactive media in education*, 2004(1), 2004.

[194] Orwant, J. Eggg: Automated programming for game generation. *IBM Systems Journal*, 39(3):782–794, 2000.

[195] Osborn, J. C., Dickinson, M., Anderson, B., Summerville, A., Denner, J., Torres, D., Wardrip-Fruin, N., and Mateas, M. Is your game generator working? evaluating gemini, an intentional generator. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 15, pages 59–65, 2019.

[196] Papert, S. Mindstorms: children, computers, and powerful ideas basic books. *Inc. New York, NY*, 10:1095592, 1980.

[197] Pears, A. and Rogalli, M. mjeliot: A tool for enhanced interactivity in programming instruction. In *Proceedings of the 11th Koli Calling International Conference on Computing Education Research*, pages 16–22, 2011.

[198] Perez-Colado, I. J., Perez-Colado, V. M., Martínez-Ortiz, I., Freire-Moran, M., and Fernández-Manjón, B. uadventure: The eadventure reboot: Combining the experience of commercial gaming tools and tailored educational tools. In *2017 IEEE Global Engineering Education Conference (EDUCON)*, pages 1755–1762, April 2017. doi: 10.1109/EDUCON.2017.7943087.

[199] Petersen, A., Craig, M., Campbell, J., and Tafliovich, A. Revisiting why students drop cs1. In *Proceedings of the 16th Koli Calling International Conference on Computing Education Research*, Koli Calling '16, page 71–80, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450347709. doi: 10.1145/2999541.2999552. URL https://doi.org/10.1145/2999541.2999552.

[200] Piaget, J. To understand is to invent: The future of education. 1973.

[201] Pillay, N. and Jugoo, V. R. An investigation into student characteristics affecting novice programming performance. *SIGCSE Bull.*, 37(4):107–110, Dec. 2005. ISSN 0097-8418. doi: 10.1145/1113847.1113888. URL http://doi.acm.org/10.1145/1113847.1113888.

[202] Piteira, M. and Costa, C. Learning computer programming: Study of difficulties in learning programming. In *Proceedings of the 2013 International Conference on Information Systems and Design of Communication*, ISDOC '13, pages 75–80, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2299-7. doi: 10.1145/2503859.2503871. URL http://doi.acm.org/10.1145/2503859.2503871.

[203] Poels, K., De Kort, Y., and Ijsselsteijn, W. " it is always a lot of fun!" exploring dimensions of digital game experience using focus group methodology. In *Proceedings of the 2007 conference on Future Play*, pages 83–89, 2007.

[204] Porayska-Pomsta, K., Anderson, K., Bernardini, S., Guldberg, K., Smith, T., Kossivaki, L., Hodgins, S., and Lowe, I. Building an intelligent, authorable serious game for autistic children and their carers. In *International Conference on Advances in Computer Entertainment Technology*, pages 456–475. Springer, 2013.

[205] Porter, L., Guzdial, M., McDowell, C., and Simon, B. Success in introductory programming: What works? *Commun. ACM*, 56(8):34–36, Aug. 2013. ISSN 0001-0782. doi: 10.1145/2492007.2492020. URL http://doi.acm.org/10.1145/2492007.2492020.

[206] Porter, L., Lee, C., Simon, B., and Guzdial, M. Preparing tomorrow's faculty to address challenges in teaching computer science. *Communications of the ACM*, 60(5):25–27, 2017.

[207] Powers, K., Gross, P., Cooper, S., McNally, M., Goldman, K. J., Proulx, V., and Carlisle, M. Tools for teaching introductory programming: what works? In *Proceedings of the 37th SIGCSE technical symposium on Computer science education*, pages 560–561, 2006.

[208] Prasad, A., Chaudhary, K., and Sharma, B. Programming skills: Visualization, interaction, home language and problem solving. *Education and Information Technologies*, 27(3):3197–3223, 2022.

[209] Prensky, M. Digital game-based learning. *Computers in Entertainment (CIE)*, 1 (1):21–21, 2003.

[210] Pérez-Colado, V. M., Pérez-Colado, I. J., Freire-Morán, M., Martínez-Ortiz, I., and Fernández-Manjón, B. uadventure: Simplifying narrative serious games development. In *2019 IEEE 19th International Conference on Advanced Learning Technologies (ICALT)*, volume 2161-377X, pages 119–123, 2019.

[211] Qian, Y. and Lehman, J. Students' misconceptions and other difficulties in introductory programming: A literature review. *ACM Trans. Comput. Educ.*, 18(1), Oct. 2017. doi: 10.1145/3077618. URL https://doi.org/10.1145/3077618.

[212] Qian, Y., Yan, P., and Zhou, M. Using data to understand difficulties of learning to program: A study with chinese middle school students. In *Proceedings of the ACM Conference on Global Computing Education*, CompEd '19, pages 185–191, New York, NY, USA, 2019. ACM. ISBN 978-1-4503-6259-7. doi: 10.1145/3300115.3309521. URL http://doi.acm.org/10.1145/3300115.3309521.

[213] Radenski, A. " python first" a lab-based digital introduction to computer science. *ACM SIGCSE Bulletin*, 38(3):197–201, 2006.

[214] Rajaravivarma, R. A games-based approach for teaching the introductory programming course. *SIGCSE Bull.*, 37(4):98–102, dec 2005. ISSN 0097-8418. doi: 10.1145/1113847.1113886. URL https://doi.org/10.1145/1113847.1113886.

[215] Rankin, Y. A. Diversity by design: Female students' perception of a spanish language learning game. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '16, pages 670–679, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4082-3. doi: 10.1145/2851581.2851610. URL http://doi.acm.org/10.1145/2851581.2851610.

[216] Rankin, Y. A., McNeal, M., Shute, M. W., and Gooch, B. User centered game design: evaluating massive multiplayer online role playing games for second language acquisition. In *Proceedings of the 2008 ACM SIGGRAPH symposium on Video games*, pages 43–49, 2008.

[217] Riese, E. and Kann, V. Training teaching assistants by offering an introductory course. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1*, pages 745–751, 2022.

[218] Riese, E., Lorås, M., Ukrop, M., and Effenberger, T. Challenges faced by teaching assistants in computer science education across europe. In *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1*, pages 547–553, 2021.

[219] Roberts, E. S. *Thinking Recursively: With Examples in Java*. John Wiley and Sons, Inc., 2005.

[220] Robins, A., Rountree, J., and Rountree, N. Learning and teaching programming: A review and discussion. *Computer science education*, 13(2):137–172, 2003.

[221] Robins, A. V. Novice programmers and introductory programming. *The Cambridge handbook of computing education research*, 1:327–376, 2019.

[222] Robinson, M. and Zinn, D. Teacher preparation for diversity at three south african universities. *Journal of Education*, 42(1):61–81, 2007.

[223] Rodrigues, G., Monteiro, A. F., and Osório, A. Introductory programming in higher education: A systematic literature review. In *Third International Computer Programming Education Conference (ICPEC 2022)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2022.

[224] Ross, J. M. Guiding students through programming puzzles: Value and examples of java game assignments. *SIGCSE Bull.*, 34(4):94–98, Dec. 2002. ISSN 0097-8418. doi: 10.1145/820127.820175. URL `http://doi.acm.org/10.1145/820127.820175`.

[225] Rouhani, M., Lillebo, M., Farshchian, V., and Divitini, M. Learning to program: an in-service teachers' perspective. In *2022 IEEE Global Engineering Education Conference (EDUCON)*, pages 123–132, 2022. doi: 10.1109/EDUCON52537.2022.9766781.

[226] Saavedra, A. B., Rodríguez, F. J. A., Arteaga, J. M. n., Salgado, R. S., and Ordoñez, C. A. C. A serious game development process using competency approach: Case study: Elementary school math. In *Proceedings of the XV International Conference on Human Computer Interaction*, Interacciand#243;n '14, pages 99:1–99:9, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2880-7. doi: 10.1145/2662253.2662352. URL `http://doi.acm.org/10.1145/2662253.2662352`.

[227] Sahami, M., Danyluk, A., Fincher, S., Fisher, K., Grossman, D., Hawthorne, E., Katz, R., LeBlanc, R., Reed, D., Roach, S., et al. Computer science curricula 2013: Curriculum guidelines for undergraduate degree programs in computer science. *Association for Computing Machinery (ACM)-IEEE Computer Society*, 2013.

[228] Sánchez-Mena, A. and Martí-Parreño, J. Teachers acceptance of educational video games: A comprehensive literature review. *Journal of e-Learning and Knowledge Society*, 13(2), 2017.

[229] Santana, B. L. and Bittencourt, R. A. Increasing motivation of cs1 non-majors through an approach contextualized by games and media. In *2018 IEEE Frontiers in Education Conference (FIE)*, pages 1–9. IEEE, 2018.

[230] Sax, L. J., Lehman, K. J., and Zavala, C. Examining the enrollment growth: Non-cs majors in cs1 courses. SIGCSE '17, page 513–518, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450346986. doi: 10.1145/3017680.3017781. URL https://doi.org/10.1145/3017680.3017781.

[231] Schanzer, E., Krishnamurthi, S., and Fisler, K. Creativity, customization, and ownership: Game design in bootstrap: Algebra. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, SIGCSE '18, page 161–166, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450351034. doi: 10.1145/3159450.3159471. URL https://doi.org/10.1145/3159450.3159471.

[232] Scherer, R., Siddiq, F., and Tondeur, J. The technology acceptance model (tam): A meta-analytic structural equation modeling approach to explaining teachers' adoption of digital technology in education. *Computers and Education*, 128: 13–35, 2019.

[233] Scholtz, T. L. and Sanders, I. Mental models of recursion: investigating students' understanding of recursion. In *Proceedings of the fifteenth annual conference on Innovation and technology in computer science education*, pages 103–107, 2010.

[234] Schulte, C. and Bennedsen, J. What do teachers teach in introductory programming? In *Proceedings of the Second International Workshop on Computing Education Research*, ICER '06, pages 17–28, New York, NY, USA, 2006. ACM. ISBN 1-59593-494-4. doi: 10.1145/1151588.1151593. URL http://doi.acm.org/10.1145/1151588.1151593.

[235] Sepchat, A., Monmarché, N., Slimane, M., and Archambault, D. Semi automatic generator of tactile video games for visually impaired children. In *International Conference on Computers for Handicapped Persons*, pages 372–379. Springer, 2006.

[236] Shahid, M., Wajid, A., Haq, K. U., Saleem, I., and Shujja, A. H. A review of gamification for learning programming fundamental. In *2019 International Conference on Innovative Computing (ICIC)*, pages 1–8. IEEE, 2019.

[237] Sharmin, S. Creativity in cs1: A literature review. *ACM Trans. Comput. Educ.*, 22(2), Nov. 2021. doi: 10.1145/3459995. URL https://doi.org/10.1145/3459995.

[238] Shein, E. Python for beginners, 2015.

[239] Simon, Luxton-Reilly, A., Ajanovski, V. V., Fouh, E., Gonsalvez, C., Leinonen, J., Parkinson, J., Poole, M., and Thota, N. Pass rates in introductory programming and in other stem disciplines. In *Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education*, ITiCSE-WGR '19, page 53–71, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450375672. doi: 10.1145/3344429.3372502. URL https://doi.org/10.1145/3344429.3372502.

[240] Slootmaker, A., Kurvers, H., Hummel, H., and Koper, R. Developing scenario-based serious games for complex cognitive skills acquisition:: Design, development and evaluation of the emergo platform. *Journal of Universal Computer Science*, 20(4):561–582, 2014.

[241] Smith, A. M. and Mateas, M. Variations forever: Flexibly generating rulesets from a sculptable design space of mini-games. In *Computational Intelligence and Games (CIG), 2010 IEEE Symposium on*, pages 273–280. Citeseer, 2010.

[242] Soloway, E., Guzdial, M., and Hay, K. E. Learner-centered design: The challenge for hci in the 21st century. *interactions*, 1(2):36–48, 1994.

[243] Sooriamurthi, R. Problems in comprehending recursion and suggested solutions. In *Proceedings of the 6th Annual Conference on Innovation and Technology in Computer Science Education*, ITiCSE '01, pages 25–28, New York, NY, USA, 2001. ACM. ISBN 1-58113-330-8. doi: 10.1145/377435.377458. URL http://doi.acm.org/10.1145/377435.377458.

[244] Sorensen, B. and Ramachandran, S. Simulation-based automated intelligent tutoring. In *Symposium on Human Interface and the Management of Information*, pages 466–474. Springer, 2007.

[245] Spiel, K., Alharthi, S. A., Cen, A. J.-l., Hammer, J., Nacke, L. E., Toups, Z. O., and Tanenbaum, T. "it started as a joke": On the design of idle games. In *Proceedings of the Annual Symposium on Computer-Human Interaction in Play*, CHI PLAY '19, pages 495–508, New York, NY, USA, 2019. ACM. ISBN 978-1-4503-6688-5. doi: 10.1145/3311350.3347180. URL http://doi.acm.org/10.1145/3311350.3347180.

[246] Sprint, G. and Fox, E. Improving student study choices in cs1 with gamification and flipped classrooms. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, pages 773–779, 2020.

[247] Squire, K. Video games in education. *Int. J. Intell. Games and Simulation*, 2(1): 49–62, 2003.

[248] Strauss, A. 8c corbin, j.(1998). basics of qualitative research: Techniques and procedures for developing grounded theory. *Thousand Oaks, CA: Sage*.

[249] Su, S., Zhang, E., Denny, P., and Giacaman, N. A game-based approach for teaching algorithms and data structures using visualizations. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, pages 1128–1134, 2021.

[250] Sweedyk, E., deLaet, M., Slattery, M. C., and Kuffner, J. Computer games and cs education: why and how. In *Proceedings of the 36th SIGCSE technical symposium on Computer science education*, pages 256–257, 2005.

[251] Sweetser, P. and Wyeth, P. Gameflow: a model for evaluating player enjoyment in games. *Computers in Entertainment (CIE)*, 3(3):3–3, 2005.

[252] Tang, S. and Hanneghan, M. A model-driven framework to support development of serious games for game-based learning. In *Developments in E-systems Engineering (DESE), 2010*, pages 95–100. IEEE, 2010.

[253] Tang, S., Hanneghan, M., and Carter, C. A platform independent game technology model for model driven serious games development. *Electronic Journal of e-Learning*, 11(1):61–79, 2013.

[254] Taylor, C., Spacco, J., Bunde, D. P., Butler, Z., Bort, H., Hovey, C. L., Maiorana, F., and Zeume, T. Propagating the adoption of cs educational innovations. In *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*, pages 217–235, 2018.

[255] Tessler, J., Beth, B., and Lin, C. Using cargo-bot to provide contextualized learning of recursion. In *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research*, ICER '13, pages 161–168, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2243-0. doi: 10.1145/2493394.2493411. URL `http://doi.acm.org/10.1145/2493394.2493411`.

[256] Theodoropoulos, A., Antoniou, A., and Lepouras, G. How do different cognitive styles affect learning programming? insights from a game-based approach in

greek schools. *ACM Trans. Comput. Educ.*, 17(1), sep 2016. doi: 10.1145/2940330. URL https://doi.org/10.1145/2940330.

[257] Thevathayan, C. and Hamilton, M. Supporting diverse novice programming cohorts through flexible and incremental visual constructivist pathways. In *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education*, ITiCSE '15, page 296–301, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450334402. doi: 10.1145/2729094.2742609. URL https://doi.org/10.1145/2729094.2742609.

[258] Tondello, G. F., Arrambide, K., Ribeiro, G., Cen, A. J.-l., and Nacke, L. E. "i don't fit into a single type": A trait model and scale of game playing preferences. In *IFIP Conference on Human-Computer Interaction*, pages 375–395. Springer, 2019.

[259] Tornero, R., Torrente, J., Moreno-Ger, P., and Manjón, B. F. E-training ds: An authoring tool for integrating portable computer science games in e-learning. In *International Conference on Web-Based Learning*, pages 259–268. Springer, 2010.

[260] Torrente, J., del Blanco, A., Cañizal, G., Moreno-Ger, P., and Fernández-Manjón, B. e-adventure3d: An open source authoring environment for 3d adventure games in education. In *Proceedings of the 2008 International Conference on Advances in Computer Entertainment Technology*, ACE '08, pages 191–194, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-393-8. doi: 10.1145/1501750.1501795. URL http://doi.acm.org/10.1145/1501750.1501795.

[261] Torrente, J., Moreno-Ger, P., Fernández-Manjón, B., and Sierra, J. L. Instructor-oriented authoring tools for educational videogames. In *2008 Eighth IEEE International Conference on Advanced Learning Technologies*, pages 516–518. IEEE, 2008.

[262] Torrente, J., Del Blanco, Á., Marchiori, E. J., Moreno-Ger, P., and Fernández-Manjón, B. ¡ e-adventure¿: Introducing educational games in the learning process. In *IEEE EDUCON 2010 Conference*, pages 1121–1126. IEEE, 2010.

[263] Torrente, J., Serrano-Laguna, A., Fisk, C., O'Brien, B., Alesky, W., Fernández-Manjón, B., and Kostkova, P. Introducing mokap: A novel approach to creating serious games. In *Proceedings of the 5th International Conference on Digital Health 2015*, DH '15, pages 17–24, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3492-1. doi: 10.1145/2750511.2750529. URL http://doi.acm.org/10.1145/2750511.2750529.

[264] Treanor, M., Blackford, B., Mateas, M., and Bogost, I. Game-o-matic: Generating videogames that represent ideas. In *Proceedings of the The third workshop on Procedural Content Generation in Games*, page 11. ACM, 2012.

[265] Vahldick, A., Mendes, A. J., and Marcelino, M. J. A review of games designed to improve introductory computer programming competencies. In *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*, pages 1–7, Oct 2014. doi: 10.1109/FIE.2014.7044114.

[266] Van Merrienboer, J. J. and Sweller, J. Cognitive load theory and complex learning: Recent developments and future directions. *Educational psychology review*, 17(2):147–177, 2005.

[267] Vasalou, A., Ingram, G., and Khaled, R. User-centered research in the early stages of a learning game. In *Proceedings of the Designing Interactive Systems Conference*, pages 116–125. ACM, 2012.

[268] Vegso, J. Continued drop in cs bachelor's degree production and enrollments as the number of new majors stabilizes. *Computing Research News*, 19(2):4, 2007.

[269] Visscher-Voerman, I. and Gustafson, K. L. Paradigms in the theory and practice of education and training design. *Educational Technology Research and Development*, 52(2):69–89, 2004.

[270] Vygotsky, L. S. *Mind in society: The development of higher psychological processes*. Harvard university press, 1980.

[271] Walker, H. M. Do computer games have a role in the computing classroom? *ACM SIGCSE Bulletin*, 35(4):18–20, 2003.

[272] Watson, C. and Li, F. W. Failure rates in introductory programming revisited. In *Proceedings of the 2014 Conference on Innovation and#38; Technology in Computer Science Education*, ITiCSE '14, pages 39–44, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2833-3. doi: 10.1145/2591708.2591749. URL `http://doi.acm.org/10.1145/2591708.2591749`.

[273] Willson, R. Analysing qualitative data: You asked them, now what to do with what they said. In *Proceedings of the 2019 Conference on Human Information Interaction and Retrieval*, CHIIR '19, pages 385–387, New York, NY, USA, 2019. ACM. ISBN 978-1-4503-6025-8. doi: 10.1145/3295750.3298964. URL `http://doi.acm.org/10.1145/3295750.3298964`.

[274] Wilson, K. A., Bedwell, W. L., Lazzara, E. H., Salas, E., Burke, C. S., Estock, J. L., Orvis, K. L., and Conkey, C. Relationships between game attributes and

learning outcomes: Review and research proposals. *Simulation and gaming*, 40 (2):217–266, 2009.

[275] Wolz, U., Barnes, T., Parberry, I., and Wick, M. Digital gaming as a vehicle for learning. In *Proceedings of the 37th SIGCSE technical symposium on Computer science education*, pages 394–395, 2006.

[276] Wu, C.-C., Dale, N. B., and Bethel, L. J. Conceptual models and cognitive learning styles in teaching recursion. In *Proceedings of the twenty-ninth SIGCSE technical symposium on Computer science education*, pages 292–296, 1998.

[277] Wu, W.-H., Hsiao, H.-C., Wu, P.-L., Lin, C.-H., and Huang, S.-H. Investigating the learning-theory foundations of game-based learning: a meta-analysis. *Journal of Computer Assisted Learning*, 28(3):265–279, 2012.

[278] Yadin, A. Reducing the dropout rate in an introductory programming course. *ACM inroads*, 2(4):71–76, 2011.

[279] Yao, L., Liu, Y., Li, W., Zhou, L., Ge, Y., Chai, J., and Sun, X. Using physiological measures to evaluate user experience of mobile applications. In *International conference on engineering psychology and cognitive ergonomics*, pages 301–310. Springer, 2014.

[280] Yarmish, G. and Kopec, D. Revisiting novice programmer errors. *ACM SIGCSE Bulletin*, 39(2):131–137, 2007.

[281] Yusoff, A., Crowder, R., Gilbert, L., and Wills, G. A conceptual framework for serious games. In *Proceedings of the 2009 Ninth IEEE International Conference on Advanced Learning Technologies*, ICALT '09, page 21–23, USA, 2009. IEEE Computer Society. ISBN 9780769537115. doi: 10.1109/ICALT.2009.19. URL `https://doi.org/10.1109/ICALT.2009.19`.

[282] Zarraonandia, T., Diaz, P., and Aedo, I. Using combinatorial creativity to support end-user design of digital games. *Multimedia Tools and Applications*, 76(6): 9073–9098, 2017.

[283] Zhang, J., Atay, M., Smith, E., Caldwell, E. R., and Jones, E. J. Using a game-like module to reinforce student understanding of recursion. In *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*, pages 1–7. IEEE, 2014.

Appendices

Appendix A

Preliminary study

A.1 Expert interview guide

Please provide your answers to the following questions as accurately as possible

1. How long have you taught introductory computer programming course at university Level?
2. What is the size of the class on average?
3. What programming language do you use?
4. What is the background of the students you teach with regard to the degree they are enrolled for and their past experience with computer programming?
5. In your rough estimate what is the failure rate in introductory programming? What are some of the factors contributing to the failure?
6. Apart from identifying and correcting errors what are the other general issues that make learning programming difficult for novices?
7. In your view which 3 topics/areas of the course syllabus do novices find most difficult to learn starting from most difficult to least difficult? Briefly explain why and where the difficulty lies.
8. What is your philosophy in terms of the order of teaching the topics?
9. What different teaching strategies/methods have you tried in the past to deal with the identified difficulties?
10. Have you evaluated the impact of your different teaching strategies?
11. If yes, what was the result? If no what are the reasons?
12. Have you tried using any game in teaching programming concept?
13. If the answer is yes, which game and what was your experience?

14. From your observation to what extent is it true that ”most university students rarely visit the university library to read or borrow a course text book to clarify any topics not under stood in class”? Please briefly elaborate.

15. In overall what recommendations would you give to improve the delivery of Introductory Programming course in higher education going forward?

A.2 Survey questionnaire

1. How many years of experience do you have in teaching introductory programming in your university?
   O 1-3 years O 3-8 years O 8-13 years O 13-20 years O Over 20 years

2. Which programming language do you use in this course?
   O C/ C++ O Java O Python O Other

3. On a scale of 1 to 5 how would you rate the following programming issues on perceived level of difficulty among novice learners?

	1-Very difficult	**2**	**3**	**4**	**5-Very easy**
Algorithms and problem solving					
Using IDE					
Identifying and correcting errors					
Understanding programming logic					
Programming syntax and semantics					
Understanding control structures					

4. On a scale of 1 to 5 how would you rate the following topics/ concepts on level of difficulty to learn among novice learners?

5. Have you ever tried difficult teaching techniques/ methods to help students learn perceived difficult topics apart from the traditional lecture method?
   O Yes O No

6. If your answer is yes to the above question, what is your response to ever trying the following teaching methods/ approaches?

7. Have you ever evaluated the impact of your different teaching approaches on learners?
   O Yes O No

	1-Very difficult	2	3	4	5-Very easy
Variables, data types and assignments					
Expressions and statements					
Input/ output handling					
Sequential structures					
Selection structures					
Looping structures					
Arrays					
Function definitions and prototypes					
Parameter passing					
Recursion					
Abstract data types and prototypes					
Records					

	Yes	No
Peer learning and support		
Using a bridging course		
Collaborative learning		
Group work		
Games - based learning (GBL)		
Referring students to the library to research on topics not clear in class		

8. If you have tried evaluating your different teaching method, what type of response did you get? .....
.....

9. If your response to Qn7 is no, what has made you not to evaluate the impact on learners? .....
.....

10. What is your level of awareness of game use in education?
O Not at all aware O Slightly aware O Some what aware O Moderately aware O Extremely aware

11. To what extent do you agree that well designed simulation games can be used to teach novices difficult introductory programming concepts?
O Strongly disagree O Disagree O Neither agree nor disagree O Agree O Strongly agree

12. From your observation, how true is the claim that “most university students rarely visit the library to read or borrow a course text book to clarify any topics not

understood in class"?
O Not very true O Not true O Neutral O True O Very true

13. In overall, what recommendations can you give to improve the delivery of introductory programming course in universities going forward?.....
.....
.....

Appendix B

User requirements study

B.1 Interview guide for teachers

I am designing a game generator tool to aid users (lecturers) automatically generate and customise educational games that can teach/ demonstrate the concept of recursion to novice students through visualization. The aim of this survey it to identify key conceptual attributes (game and learning attributes) necessary to stimulate the design and development of prototype of such a tool.

B.1.1 Teaching recursion experience

1. How long have you taught introductory computer programming course in university?
2. How much time do you dedicate to the recursion topic?
3. What content do you teach novices in recursion, and in what order?
4. Have you used games to teach introductory programming concepts in class?
5. What are the potential barriers to the adoption of games approach in teaching?

B.1.2 Recursion educational games and game generator tool design requirements

1. What game features should I consider during design of my tool for the games to be generated to teach novices (beginner students) recursion?
2. What pedagogical/ instructional design considerations do you recommend for the games generated by the authoring tool to render them useful in teaching recursion?
3. What teaching examples do you use in class for recursion?
4. What examples/ tasks / game analogies / real life scenarios/ do you recommend to be designed for the games generated by the tool to allow novices visualize

both active and passive flows of recursion? Kindly provide sketches of paper prototypes.

5. What aspects of recursive program execution steps should I design in my games to make beginner programming students visualise/ see the concept of recursion? Kindly provide sketches of paper prototypes.

Appendix C

Experiments 1, 2 , 3, and 4 Questionnaires and Ethics Clearance)

C.1 Experiment 1 (Pre-test questionnaire)

1. Which of the following best describes your age group?
   O Below 25 years O 20 -30 years O 31-35 years O 36-40 years O 41-45 years O 46 -50 years O Over 50 years

2. What is the highest degree or level of education you have completed?
   O Secondary level certificate O Bachelor degree O Masters degree O PhD. or Higher

3. Which one of the following best describes the institution where you work?
   O High school O Tertiary college O University

4. Please rate your teaching experience in years. Experience here also includes High school teacher/ course tutor/ teaching assistant [TA] / lecturer position in a higher education setting.
   O Less than 5 years O 5 -10 years O 11-15 years O 16-20 years O Over 20 years

5. Which Computer Science (CS) programming courses do you teach or have taught before?
   O Only Fundamentals of programming (CS0)/ High schools/ College courses
   O Only Introductory programming (CS1) Year 1 programming courses
   O Only Computer Science 2 (CS2)/ Year 2 programming courses
   O Both CS1 and CS2
   O Programming courses in other years (OOP in Yr3)
   O CS1, CS2 and OOP in YR3

6. Which of the following programming languages is used to teach fundamentals of programming / introductory programming course in your institution?
   O Python
   O C

O C++
O Java
O Other

7. Do you consider yourself a gamer?
   O Yes O No

8. If yes, how often do you play video games for leisure in a month?

   O Daily O 2 to 3 three times a week O Weekly O After 2 weeks O After 3 weeks

9. If no, why? .....

10. Have you ever had any experience of teaching programming using games?
    O Yes O No

C.2 Experiment 1 (Post-test questionnaire)

1. Which Operating System did you use to test the tool?
   O Windows
   O Linux
   O MAC
   O Other

2. On a scale of 1 to 7, overall please rate how easy or difficult you found each of the following tasks to be?

	1-Very difficult	**2**	**3**	**4**	**5**	**6**	**7-Very easy**
Task 1: Creating my user account and logging into the system							
Task 2: Creating a custom game using a given example by following a few steps							
Task 3: Customizing the generated game							
Task 4: Downloading the created game							

3. On a scale of 1 to 7, overall please rate how easy or difficult you found each of the following tasks to be?

	1-Very difficult	**2**	**3**	**4**	**5**	**6**	**7-Very easy**
Task 1: Creating my user account and logging into the system							
Task 2: Creating a custom game using a given example by following a few steps							
Task 3: Customizing the generated game							
Task 4: Downloading the created game							

4. On a scale of 1 to 7, overall please rate how easy or difficult you found each of the following tasks to be?

	1-Very difficult	**2**	**3**	**4**	**5**	**6**	**7-Very easy**
Task 1: Creating my user account and logging into the system							
Task 2: Creating a custom game using a given example by following a few steps							
Task 3: Customizing the generated game							
Task 4: Downloading the created game							

C.3 Experiment3 Questionnaire - Part A: Demographics

In this survey you will provide your demographic information and your final free comments about your experience with the proposed game generator tool.

Part A: Demographic Information

1. Which of the following best describes your age in years?
O Below 18 years
O 18 to 19 years
O 20 to 21 years
O 22 to 23 years
O 24 to 25 years
O Over 25 years

2. What are your 2 teaching subjects?
O Computers and Maths
O Computers and Physics
O Computers and Chemistry
O Computers and Business studies O Other

3. Do you consider yourself a gamer?
O Yes
O No

4. How would you rate your skills in game programming?
O Non at all
O Low
O High

5. Did you study Computers in Secondary School?
O Yes
O No

6. Have you been taught any programming course using games?
O Yes
O No

C.4 Experiment3 Questionnaire - Part B: User experience when creating a game with a static game example (Prototype with a static game example)

Please provide your impressions of the Recursive Game Generator Tool after creating and playing a dynamic custom game by check marking your impression on a scale of 1 – 7 between the terms offered in each line below.

		1	2	3	4	5	6	7	
Pragmatic Quality (PRA)	Technical								Human-centred
	Complicated								Simple
	Impractical								Practical
	Cumbersome								Straightforward
	Unpredictable								Predictable
	Confusing								Clearly structured
	Unruly								Manageable
Hedonic Quality Identity (HQ-I)	Isolating								Connective
	Unprofessional								Professional
	Tacky								Stylish
	Cheap								Premium
	Alienating								Integrating
	Separates me from people								Brings me closer to people
	Unpresentable								Presentable
Hedonic Quality Stimulation (HQ-S)	Conventional								Inventive
	Unimaginative								Creative
	Cautious								Bold
	Conservative								Innovative
	Dull								Captivating
	Undemanding								Challenging
	Ordinary								Novel
Attractiveness (ATT)	Unpleasant								Pleasant
	Ugly								Attractive
	Disagreeable								Inviting
	Rejecting								Pleasant
	Bad								Good
	Discouraging								Motivating

D: Free text questions

1. What are some of the bugs or technical issues you encountered during the usability test? ----------
   ----
2. Please freely provide any further overall opinions/ advice about the Recursive Game Generator tool? ----------
   ----

C.5 Experiment3 Questionnaire - Part C: User experience when creating a game with a dynamic game example (Prototype with a dynamic game example

Please provide your impressions of the Recursive Game Generator Tool after creating and playing a dynamic custom game by check marking your impression on a scale of 1 – 7 between the terms offered in each line below.

		1	2	3	4	5	6	7	
Pragmatic Quality (PRA)	Technical								Human-centred
	Complicated								Simple
	Impractical								Practical
	Cumbersome								Straightforward
	Unpredictable								Predictable
	Confusing								Clearly structured
	Unruly								Manageable
Hedonic Quality Identity (HQ-I)	Isolating								Connective
	Unprofessional								Professional
	Tacky								Stylish
	Cheap								Premium
	Alienating								Integrating
	Separates me from people								Brings me closer to people
	Unpresentable								Presentable
Hedonic Quality Stimulation (HQ-S)	Conventional								Inventive
	Unimaginative								Creative
	Cautious								Bold
	Conservative								Innovative
	Dull								Captivating
	Undemanding								Challenging
	Ordinary								Novel
Attractiveness (ATT)	Unpleasant								Pleasant
	Ugly								Attractive
	Disagreeable								Inviting
	Rejecting								Pleasant
	Bad								Good
	Discouraging								Motivating

D: Free text questions

1. What are some of the bugs or technical issues you encountered during the usability test? ----------
   ----
2. Please freely provide any further overall opinions/ advice about the Recursive Game Generator tool? ----------
   ----

C.6 Questionnaire for Experiment 4 with CS1 Students

Part A: Demographic information

1. How old are you?_____

2. Are you majoring in Computer Science?
O Yes
O No

3. How would you rate yourself when it comes to programming in Python?
O Very poor O Poor O Neutral O Good O Very good

4. Do you consider yourself a gamer? (A gamer is somebody who likes playing computer games)
O Yes
O No

5. Have you been taught using games before?
O Yes
O No

6. Which game did you play?
O Mushroom picker game
O DnD game

C.7 Experiment 4 Game Experience Questionnaire

Part B: Game Experience (Game Experience Questionnaire)

Please indicate how you felt while playing the given generated game for each of the following items on the scale: 0 - Not at all, 1 - Slightly, 2 - Moderately, 3 - Fairly, 4 - Extremely.

		0 – Not at all	1	2	3	4- Extremely
1	I felt content					
2	I felt skilful					
3	I was interested in the game story					
4	I thought it was fun					
5	I was occupied with the game					
6	I felt happy					
7	It gave me a bad mood					
8	I thought about other things					
9	I found it tiresome					
10	I felt competent					
11	I thought it was hard					
12	It was aesthetically pleasing					
13	I forgot everything around me					
14	I felt good					
15	I was good at it					
16	I felt bored					
17	I felt successful					
18	I felt imaginative					
19	I felt that I could explore things					
20	I enjoyed it					
21	I was fast at reaching the game's targets					
22	I felt annoyed					
23	I felt pressured					
24	I felt irritable					
25	I lost track of time					
26	I felt challenged					
27	I found it impressive					
28	I was deeply concentrated in the game					
29	I felt frustrated					
30	I felt like a rich experience					
31	I lost connection with the outside world					
32	I felt time pressure					
33	I had to put a lot of effort into it					

C.8 Experiment4 Questionnaire - Parts C , D and E

Part C: Perceptions of Learning Value and Enjoyment (Adapted from [3])

On a scale of 1 to 5, please rate your perceptions of the learning value and enjoyment of the played Mushroom Picker game to the extent that you disagree or agree to the following statements.

		1-SD	2	3	4	5-SA
1	I found practicing coding using a Python IDE in the Mushroom game environment a motivating way of learning programming					
2	I found the mushroom analogy/ metaphor chosen for the Mushroom Picker game appropriate for learning the recursion concept					
3	I found visualization in the Mushroom Picker Game a better way of learning recursion					
4	The Mushroom Picker game played was effective for leaning the concept of recursion					

On a scale of 1 to 5, please rate your perceptions of the learning value and enjoyment of the played DnD game to the extent that you disagree or agree to the following statements.

		1-SD	2	3	4	5-SA
1	I found practicing coding using a Python IDE in the DnD game environment a motivating way of learning programming					
2	I found the swing sword analogy/ metaphor chosen for the DnD game appropriate for learning the recursion concept.					
3	I found the factorial analogy chosen for the DnD game appropriate for learning the recursion concept					
4	The DnD game played was effective in learning the concept of recursion					

Part D: Feature Usefulness (Adapted from [2,4])

Overall, what game interface design features did you find most useful?

☐ Python Integrated Development Environment (IDE)

☐ Programming tasks

☐ Character customisation

☐ Player reward and motivation (through health points and unlocking trophies)

☐ Color use and design layout

☐ Multimedia elements

☐ Navigations and interactions

Other _____

Part E: Open Ended questions (Adapted from [2,3]

1. What changes would you like to see in the generated games to improve their effectiveness for learning?

2. What is your final comment about the generated games?

C.9 Experiment 1 Positive Comments Open Ended Responses Raw Data

	What three things did you like most with the game generator tool		
Participant ID	**#1 like**	**#2 like**	**#3 like**
1	There were no technical issues. The encounter was amazing.	The generator tool prototype is a convenient tool for generating games, especially for teachers to use as a teaching aid.	
2	Easy to use	It's fast and reliable project based learning	User-friendly
3	Make explanation of concepts	Abit easy to use	Easy to access help
4	Its practicabilty	The ease of use	Quick response on the difference action
5	if incorrect, option to play again	error messages on retun	display of both the game and IDE
6	Automatic checking of errors	Trophies won by students	Customization
7	The coding test for students	The game generator	Easy to get help
8	Teaching recursive topic	opportunity to customise the game	different asset in the game
9	gives the learner an opportunity try out tasks	ability to try different options	chance to repeat a task and seek help
10			
11	Simplicity	User interface	Instructions
12	Easy to use	doesn't require programming skills	Easy to download
13	The ability to create my own games	The trophy and point incentives to motivate learners	The proper labels and description during game development
14	the tool is user friendly	easy to use	very well elaborated steps
15	Could add fun to doing some programming	No	No
16	Easy to use	Short steps	rewarding
17	It is a fun and effective way to teach Recursion in Computer Programming.	It is user friendly.	Easy to use.
18	easy to use	fun	easy to generate games
19	It takes someone step by step on how to do the task	It can Motivate student to learn on there own	it promotes Creativity and exploratory learning
20	Allowing users to customize actors	Ability to downlod the game	Ability to write simple codes
21	I think it is a really fun way to teach from basic to more advanced programming concepts	One problem lecturers face with teaching programming is the abstract nature of programming constructs. THis game does a lovely job of making the constructs visual and concrete for learners	I just thought it was very easy to use, very quick to set up and that it would be fun to do in class
22	Login and account creation feature seems solid and well-implemented.	Game generator tool is relatively minimalistic, which I like	It gives the option of creating a game from an existing template quite quickly
23	easy to implement	easy to learn	good interface
24	Generation feature	Customisation feature	The idea is great
25	Customisation	Trophy	Programming interphase
26	It is simple to use	Easy to customize	very interactive
27	creating game from scratch	Generating different kinds of games	customizing game
28	user friendly	easy to use	teaches recursion well
29	game generation	learning motivation rewards	learning support comments
30	Integration of programming interface	Health points award for task completed successfully	Penalty/loss of points for tasks not completed successfully

C.10 Experiment 1 Negative Comments Open Ended Responses Raw Data

	What three things did you Not like most with the tool		
Participant ID	**#1 di-slike**	**#2 dis-like**	**#3 dis-like**
1	Logging in and verification of code	Require skilled personnel	More tool should be added
2	Needs a lot of time to understand	Difficult	Help not very comprehensive
3	It doesn't allow full programming by the students	Not flexible in changing the program outcome	None
4	not sure of any	trophies	plays in one domain -- perhaps need like league, championship or etc. not sure if the levels in the game
5	Language options not available	Errors handled are limited	Limited Coverage
6	Iinvolving too many steps	The game loading took some time to complete because of slow internet speed	Initially difficult steps
7	quality of the 3D experience is poor	not so engaging	takes time to learn
8	not all learners learner phyton	none	none
9	its based on recursion; which is somewhat an advanced concept of programming.	x	x
10	Very poor feedback	Need to give better feedback as levels progress	Not informative enough on progressive levels
11	The issue of log in before use	The tool should be very important if physical class is conducted before use	Recursion is an advanced topic for novice learners
12	The interface in not intuitive	Lack of support of other operating systems	The authentication process is too strict for a teaching tool
13	n/a	n/a	n/a
14	Instructions to students in the given examples is really fuzzy	Given examples are not informative and to my opninion not always correct	Could not figure out how I can create my own games
15	n/a	n/a	n/s
16	None.	N/A.	N/A.
17	not easy how to figure how to use it in the first place	the code in the IDE is somehow not easy to figure its purpose	the ide should state clearly that it expects one to write python code using python conventions like
18	It take abit of time to generate the game	It requires some one to follow all steps	does not provide hints on how to solve the problem
19	static movement	Dwonload time for the game	Incoporation of audi file
20	The UI(sorry)	I think that it might be a bit of a miss for girls - we often forget about females when we think programming	Nohting else
21	NA	NA	NA
22	Game customization feature is not well-explained at all (especially the automarker and code completion sections for the students), not even by the video. The tutorial video is also far too long	Lack of ability to customise more parts of it (for example, the editor, language (cannot even choose Python 3, which is more recent), and the general flow of it)	Help section is confusing and it takes effort to notice and remember relevant parts
23	easy to apply	good	can be used to teach even low level students
24	User Interface	Auto marker feature vague	Registration feature
25	I expect to see the Sprite show what the program is doing or how the recursion happens.	Too much steps to follow	I only observed the two points mention above
26	A number of steps to log in	Loosing health points on wrong tasks	No comments
27	loging in is problematic	student may be carried away by entertainment aspect of games	Creation of games requires several trials
28	could improve the pictures to be	not applicable	not applicable
29	charcter animation not clear	user interface is too plain	lack of debugging suggestions
30	Use of python, is not common in introductory to programming classes	Dwelling on one concept, that is recursion	Downloading and setting up the tool, it should run from cloud

C.11 Experiment 1 Final Comments Raw Data

Participant ID	Final Comments
1	There were no technical issues. The encounter was amazing.
2	A Good tool for teaching programming
3	Wonderful tool... Great
4	It is a good tool that forms an excellent basis for game programming
5	on validating the code, show display other ways to code the problem. i feel like it validate the code basing on programmers syntax only. if the programmer can provide other
6	It would be a good tool if completed
7	It's a good teaching and learning tool for programming language teachers and students
8	Great! has the potential to support learning CS
9	its a good tool for teaching pogramming at lower levels
10	should be expanded to support other programming concepts
11	It is a good tool. However, more needs to be done on the feedback mechanisms of the tool. Consider telling the student what is wrong with the code in the way better than an IDE does, or like what an IDE does.
12	The tool simplifies programming to a great extent
13	Its an excellent tool that will definately improve learning gains
14	perfect tool for teaching programming.
15	This seems to be an interface to an existing auto-grader. It is not all clear how the auto
16	Its a useful learning tool to us lecturer and students.
17	I strongly recommend it for adoption in teaching Recursion in Computer Programming.
18	its a brilliant tool for teaching the concept of functions and recursion in particular.
19	Good for teaching Programming at an introductory level
20	The research is very useful and it will enable our students to learn how to use such
21	I think it is lovely and a lot of fun - I think it would help a lot of students and lecturers and maybe entice some of the lazier kids to get involved.
22	Overall, I don't think that the games that this tool generates would teach students recursion more effectively than the usual assignments given in Computer Science courses. To be blunt, I think this is just a nice way to "gamify" any programming task, not just recursion. There are actually games out there on the market, that I have played, that teach much more about the concepts behind recursion than this tool, unfortunately. Also, I noticed quite a few spelling mistakes and strange behaviours in the generated games (for example, we don't automatically go to the next level when finishing a level), but I don't want to go into too much detail here. I feel that the creator of this tool must definitely do some more research on current games on the market that can teach programming (since these would also be available to universities, for a price), as well as academic literature perhaps (although I am not well-versed in this area of Computer Science at all) and make design decisions for the Recursive Game Generator Tool informed by what is currently state of the art.
23	its a very good tool
24	The User Interface should be improved, automarker should be well explained, put practical examples
25	It will much appealing to introductory programming students if the game is not just about text based programming in recursion but as well show how the it happens in graphic form.
26	I highly recommend the use of this tool in teaching programming
27	its very applicable in programming
28	it is good
29	This is a great tool that can help students to connect class concepts with real applicability.I would recommend it to be improved further, and adopted for teaching.
30	A very good tool

C.12 Experiment 2 Open Ended Responses Raw Data

Participant ID	Positive and Negative and Suggestion for improvement	Final Comments
1		May be useful but the current user experience in getting started reduces the potential value
2		Cool, good work!
3		The tool is good and I strongly recommend it.
4	Interface	The tool is useful and the idea thereof
5	There is no feedback if there are errors - e.g. are they syntactic or semantic? could highlighting of the error line for the former be included? Or could the basic syntax of the IF statement be displayed somewhere whenever an IF is required for example? For D&D you should give the value of 0! rather than say output 0! ie rather say output 1. It would be nice if some "well done	The best part is the way it has students gradually build up the coding of a recursive solution. The advantage of that cannot be overemphasised - well done!
6	Character customization Reward scheme	It can be helpful in engaging students in programming classes
7	It is friendly	It enhances creativity
8	None	Easy to use
9	Very easy and interesting to use	leads to faster game generation
10	It's web based, would wish it's offline	A very interesting app for teaching ICT students
11		It is a good invention
12	Web designing	Requires an orientation for Teachers because its the best tool for teaching.
13		Its the best tool. And will recommend to my colleague it will help to produce students who know how to do things.thnks to the team for the thought
14	It was interesting	Good game recommended for teaching
15		Its a wonderful tool for learning
16	Not applicable	Until I use it
17	It is interesting	It is of important use
18	Being web based, means only useful when and where there is internet, perhaps costly to some	Very good item. Much needed in our systems!!
19	It's userfriendly	It's a an interesting tool
20		It is a good and helpful
21	IDE looks dull. Could have integrated a more attractive IDE e.d. Visual code, Atom, Sublime, Jupyter Notebook. These IDE support you as you type in terms of code completion or even highlight errors in the code, instead of waiting to see the error on run time.	Happy with the whole idea of gamification in education that's why I am willing to share this with my gaming students.
22		It is a promising tool and the future customization of games and their problems will be fantastic. However, I would like to call attention to some of its characteristics. 1) The Mushroompicker is great. It abe used to help students to understand LOOP also. In the case of the last question (regarding recursion), as the students can not see all the paths and their obstacles, they may have some problems reaching a good solution. 2) In my opinion, students may experience a cognitive load on trying to understand what they have to do to solve the Runner game.
23	Easy and could save time	Could be very useful for teaching.
24	The generated package is rather large, over 70 MB. But I guess that with Unity there is not much to do as the library itself is large.	The tool seems to create the games as expected. It is simple but provides the essential aids.
25	Log in feature	The built-in template
26	None.	It is a timely and very useful game generator tool that makes it easier for me to create customised games to assist in teaching the topic of recursion.Its Help, built-in game generation template, Custom game generation, game download and distribution support features stand out in making it very easy to learn and use. I also like the fact that it has support for common operating systems (Windows and Linux).I highly recommend it for widespread distribution and use.
27	Optimizes on available resources to improve learning and teaching skills	Worth to be adopted by higher learning institution in the face of changing technological innovation
28	Usage and zero-programming	Interface gas to improve

C.13 Experiment 3 Open Ended Responses Raw Data

Participant ID	Bugs or Technical issues	Final Comments/ Opinions and Recommendations for Improvemenet
3	There were no technical issues. The encounter was amazing.	The generator tool prototype is a convenient tool for generating games, especially for teachers to use as a teaching aid.
5		
7	none	The game should be more presentable and realistic. use real and attractive features
8	knowing the exact manner of inputting the solutions was tricky and how to indent to in the right way was hard too .	the game looks interesting and fun to play ,easy to generate but a clear procedure should be put in place on how to indent and place the solutions.
9	none at all	I recommend the game
10		
11	Failure to run a program due to some of my errors in my program	Increases and establish our programming skills Can be used by teachers in teaching because first it is more fun, Challenges learners to be also innovative in the field of developing games Good for teaching because it is more interactive between the teacher and learners
12	Slow internet	
13	Low network connections	It can help students and teachers in programming
14	network server buffer	it is great and enjoyable
15	There were no issues encountered.	The game generator tool is a noble idea since it aids teachers to teach programming.
16	I didn't encounter any problems during the test.	Yes its a good idea.
17	None at all	This is a good generator tool for teachers even though they dont have the skills to generate. Therefore it is
18		
19	Indentation was somehow challenging.	It is a very interactive game and practical making learning easy
20		
21	Network speed was very low.	It is an interesting game which boost our speed in gaming.It was a good experience.
23	no technical issues	the games are just good
24	syntax errors	its usable to teach
25	no technical issues	yes i agree
26	slow internet	The creation of the game was good and a learning experience at the the same time. More practices should be done for familiarity.
29	Actually i didn't have any bugs the functioning of the program was very straight-forwad and precise	I'd advice on working with providing different colors for the conditionals and statements as it will most likely help the students to keep track of their code efficiently especially when it comes to modifications and debugging, it will help to reduce the plain looking codes that are all in grey. Otherwise i like the game and i think it will provide a good platform for game developer wannabes. I'm not much of a game developer but seeing that helped paint a picture of the whole process while it also makes the process look simple. Keep up the good work.

C.14 Experiment 4 Open Ended Responses Raw Data

Participant ID	Suggestion for improvement	Final Comments
1	Do not limit the games to recursion, teach almost every module using games or real world application	They were fun, and challenged my knowledge on recursion
2	change. It would be great if when you enter a piece of code, some change would happen in the game, without having to write the full code and realise at the end that the code has bugs.	They were a bit difficult to understand. It was hard to determine what to do taking into account that I don't know recursion. They assumed some knowledge of recursion.
3	I do not believe that the analogy used for the D and D game was effective. If compared to the Mushroom Picker game, where there was a clear set of steps that needed to be repeated, the D and D game was harder to visualize. I think that a more fitting analogy would greatly improve the learning experience of the D and D game. For example, something like a car being required to move around a race track for a specified number of laps would have been more fitting.	When I was initially taught Recursion in CSC1015F, I found it extremely hard to understand due to the fact it was pretty unconventional compared to standard looping mechanisms. However, in the way that the Mushroom Picker game was presented, I felt it was much easier to grasp the concepts being taught. Furthermore, the gradual approach of the games was extremely effective in comparison to being taught the entire section in a few days with few physical examples.
4		It was beneficial with regards to the visualisation of concepts relating to recursion and developing and understanding of the types of situations recursion in useful for.
5	The games are fine for learning, though I felt I wanted more. A little more challenge would be nice or rather extra stages.	I enjoyed the games, especially the DnD game. Adventure is a good genre for learning and it matched the concept (recursion) nicely. I look forward to another
6	A more user friendly IDE Clearer instructions Good Color use to increase convenience for the user More demonstrations and examples Pseudocode hints	The idea to use games to learn programming is a good one and with good thinking behind it would help a lot of beginner programmers. However programming will be programming if the students are not exposed to algorithms and pseudocode and just told to pop them out.
7	There should be more of a build up as in DnD to a whole recursion solution. I also feel as though the tasks were not very clearly and it would be nicer to get some sort of feedback when things go wrong otherwise it is frustrating. Maybe a leaderboard could be nice to motivate people too. The DnD game was very nice to play to learn recursion as it asked for tasks that made sense and there was a gradual build up to a full recursive call that we had to do ourselves.	I think the concept is cool and it would be nice to learn recursion using interactive games. I think it helped me understand the concept better and it was easier to understand through visuals and working through a game.
8		Very strategic approach and with the right type of games t could help a lot.
9		The games were appropriately content- related and quite fun. The DnD game did not have a visual layout so I could not understand fully what was expected of me since the method I used was incorrect.
10	Possibly more guidance when approaching a new level.	I really enjoyed them. They were quite helpful to a large degree and make learning programming a lot easier. I would suggest starting easier though for beginners, however I am aware this was meant for people with some kind of python experience.
11	Resources within the program to show you how the coding works. Such as how to access the 2D array and iterate through it	The games didn't actually teach anything. They were more of a test of my skills. There was no information or way to learn about recursion.
12		It helped a lot in learning programming as it made it a bit simpler to learn.
13	I think the tutorials should be more in depth. I really struggled with the Mushroom Game. To me the first level's code was too complex and I struggled. The IDE was also not ideal in that game. It was however very suited to the DnD game. The DnD game had me whizz through the levels because the explanations and manageable goals really helped.	Overall, I enjoyed the experience. I disliked the Mushroom game and think it's too much for a novice but I enjoy the DnD game with it's explanations and bite sized code. It really made recursion easier, when it's taught step by step like that. I could see myself getting into coding as a hobby if it was taught with games. I hope this helps with your research.
14	I would like to see more interesting and realistic analogies than the swinging of the sword.	The games were very fun and provided a much more enjoyable and different learning experience which I think should be adopted here and there.
15	Better visualisation of the running code would help. It felt like you needed to have an understanding of recursion to know what was happening. The code pre-populated in the IDE had formatting that wasn't accepted by the compiler. I had to re-write a lot of it to get the code to work.	I think there is potential for using games to teach concepts like recursion, but these did not have the visualisation requisite to clarify the concept.
16		This games are way too good and I think it's time kids play those kind of games to learn programming
17		The concepts were effective implantations for learning recursion, however the poorly-worded instructions, confusing layout and overall frustrating UX made it incredibly difficult to get to the parts of the games that actually worked.
18	If I were being introduced to recursion for the first time through these games, I would have been completely lost. The games and goals were not explained well, and I was confused, specifically with the DnD game, about what we were and were not allowed to add or return. It would help if there was a visualised and explained solution that came up as optional after a few attempts at the level. Otherwise, it becomes really frustrating. The mushroom picker would benefit from a guided tutorial- knowing what solution exactly the game wanted, although I had "picked" (to my understanding) the	The mushroom picker game, if combined with story like the DnD game, has the most potential to be useful as a tool for learning recursion. The visualisation was helpful to a lot of us when we watched videos to understand programming concepts like these, so it definitely is something students would appreciate if implemented correctly.
19		I think that the fact that it was so interactive and visual is what helped me better understand the concepts of recursion

C.15 Ethics Approval for Preliminary Needs Assessment Study

UNIVERSITY OF CAPE TOWN
IYUNIVESITHI YASEKAPA • UNIVERSITEIT VAN KAAPSTAD

Faculty of Science
University of Cape Town
Rondebosch
South Africa 7701

Tel: +27 21 650 2866/7
E-mail: mdensmore@cs.uct.ac.za

4 July 2018

Jecton Anyango
Department of Computer Science

RE: Teaching Novices Recursion Using Simulation Game

Dear Jecton Anyango

I am pleased to inform you that the Faculty of Science Research Ethics Committee has approved the above-named application for research ethics clearance, subject to the conditions listed below.

- Implement the measures described in your application to ensure that the process of your research is ethically sound; and
- Uphold ethical principles throughout all stages of the research, responding appropriately to unanticipated issues: please contact me if you need advice on ethical issues that arise.

Your approval code is: **FSREC 48 - 2018**

I wish you success in your research.

Yours sincerely

Dr Melissa Densmore
Acting Chair: Faculty of Science Research Ethics Committee

Cc: A/Prof Hussein Suleman (Supervisor)

C.16 University of Cape Town Ethical Clearance

UNIVERSITY OF CAPE TOWN
IYUNIVESITHI YASEKAPA • UNIVERSITEIT VAN KAAPSTAD

Faculty of Science
University of Cape Town
Rondebosch
South Africa 7701

E-mail: shari.daya@uct.ac.za
Tel: 021 650-2880

29 May 2019

Jecton Anyango
Department Computer Science

RE: An Investigation of a Game Generator Tool to Teach Recursion

Dear Jecton Anyango

I am pleased to inform you that the Faculty of Science Research Ethics Committee has approved the above-named application for research ethics clearance, subject to the conditions listed below.

- Implement the measures described in your application to ensure that the process of your research is ethically sound; and
- Uphold ethical principles throughout all stages of the research, responding appropriately to unanticipated issues: please contact me if you need advice on ethical issues that arise.

Your approval code is: **FSREC 48 - 2019**

I wish you success in your research.

Yours sincerely

Dr Shari Daya
Chair: Faculty of Science Research Ethics Committee

Cc: Associate Professor Hussein Suleman (supervisor)

C.17 The National Commission for Science, Technology and Innovation (Nacosti) Research Permit

REPUBLIC OF KENYA

NATIONAL COMMISSION FOR SCIENCE,TECHNOLOGY & INNOVATION

Ref No: **561690**

Date of Issue: **14/January/2021**

RESEARCH LICENSE

This is to Certify that Mr.. Jecton Tocho Anyango of University of Cape Town, has been licensed to conduct research in Nairobi on the topic: An Investigation of a Game Generator Tool to Teach Recursion for the period ending : 14/January/2022.

License No: **NACOSTI/P/21/8402**

561690

Applicant Identification Number

Director General
NATIONAL COMMISSION FOR SCIENCE,TECHNOLOGY & INNOVATION

Verification QR Code

NOTE: This is a computer generated License. To verify the authenticity of this document, Scan the QR Code using QR scanner application.

C.18 Kenyatta University Permission to Conduct Research

KENYATTA UNIVERSITY

OFFICE OF DEPUTY VICE-CHANCELLOR, RESEARCH, INNOVATION AND OUTREACH

Ref: KU/DVCR/RCR/VOL.3/309

P. O. Box 43844 – 00100
Nairobi. Kenya
Tel. 254-20-810901 Ext. 026
E-mail: dvc-rio@ku.ac.ke

Mr. Jecton Anyango ,
Department of Computer Science,
University of Cape Town,
SOUTH AFRICA

25th January, 2021

Dear Mr. Anyango,

RE: **REQUEST TO COLLECT RESEARCH DATA AT KENYATTA UNIVERSITY**

This is in reference to your letter dated 21st January, 2021 requesting for authorization to collect research data at Kenyatta University on the topic **"An Investigation of a Game Generator Tool to Teach Recursion"** *towards the award of a PhD degree of* University of Cape Town.

I am happy to inform you that the Vice-Chancellor has approved your request to collect data. It has been noted that your data will be collected from the group of Bachelor of Education students in the Department of Computing & Information Tecnology.

The University requires that, upon completion of your research, you submit a hard copy of your thesis to the Deputy Vice-Chancellor, Research who shall forward it to the University Library. Kindly therefore download, complete and sign Form RIO3 and return it to my office prior to the commencement of

C.19 Ethics Approval to Conduct Research with UCT CS1 Students

UNIVERSITY OF CAPE TOWN
IYUNIVESITHI YASEKAPA • UNIVERSITEIT VAN KAAPSTAD

Faculty of Science
University of Cape Town
Rondebosch
South Africa 7701

E-mail: melissa.densmore@uct.ac.za
Tel: 021 650-9111

16 June 2021

Jecton Anyango
Department of Computer` Science

An Investigation of a Game Generator Tool to Teach Recursion

Dear Jecton Anyango

I am pleased to inform you that the Faculty of Science Research Ethics Committee has approved the above-named application for research ethics clearance, subject to the conditions listed below.

- Clearance has NOT been given for in-person research
- Restrictions on involving human participants in research must be adhered to, given current concerns about the spread of Covid-19. Please ensure that you are aware of and comply with UCT policy on this, as communicated by management.
- Implement the measures described in your application to ensure that the process of your research is ethically sound; and
- Uphold ethical principles throughout all stages of the research, responding appropriately to unanticipated issues: please contact me if you need advice on ethical issues that arise.

Your approval code is: **FSREC 055 – 2021**

I wish you success in your research.

Yours sincerely

Yours sincerely

Dr Melissa Densmore
Acting Chair: Faculty of Science Research Ethics Committee

C.20 Ethics Approval to Access UCT Students

	RESEARCH ACCESS TO STUDENTS	DSA 100

NOTES

1. This form must be **FULLY** completed by all applicants who want to access UCT students for the purpose of research or surveys.
2. Return the fully completed **(a) DSA 100** application form **by email**, in **the same word format**, together with **your: (b) research proposal inclusive of your survey, (c) copy of your ethics approval letter / proof (d) informed consent letter** to: Nadierah.Pienaar@uct.ac.za. You application will be attended to by the Executive Director, Department of Student Affairs (DSA), UCT.
3. The turnaround time for a reply is **approximately 10 working days**.
4. NB: It is the responsibility of the researcher/s to apply for and to obtain **ethics approval and to comply with amendments that may be requested;** as well as **to obtain** approval to access UCT staff and/or UCT students, from the following, at UCT, respectively: (a) **Ethics**: Chairperson, Faculty Research Ethics Committee' (FREC) for ethics approval, (b) **Staff access:** Executive Director: HR for approval to access UCT staff, and (c) **Student access:** Executive Director: Student Affairs for approval to access UCT students.
5. **Note:** UCT Senate Research Protocols requires compliance to the above, **even if prior approval has been obtained from any other institution/agency. UCT's research protocol requirements applies to *all* persons, institutions and agencies from UCT and external to UCT who want to conduct research on human subjects for academic, marketing or service related reasons at UCT.**
6. **Should approval be granted to access UCT students for this research study, such approval is effective for a period of one year from the date of approval (as stated in Section D of this form), and the approval expires automatically on the last day.**
7. **The approving authority reserves the right to revoke an approval based on reasonable grounds and/or new information.**

SECTION A: RESEARCH APPLICANT/S DETAILS

Position	Staff / Student No	Title and Name	Contact Details (Email / Cell / land line)
A.1 Student Number	ANYJEC001	Mr. Jecton Anyango	anyjec001@myuct.ac.za / +27-23-56-1915 / +254-722492848
A.2 Academic / PASS Staff No.			
A.3 Visitor/ Researcher ID No.			
A.4 University at which a student or employee		Address if *not* UCT:	
A.5 Faculty/ Department/School	Faculty of Science / Department of Computer Science		
A.6 APPLICANTS DETAILS If different from above	Title and Name	Tel.	Email

SECTION B: RESEARCHER/S SUPERVISOR/S DETAILS

Position	Title and Name	Tel.	Email
B.1 Supervisor	Prof. Hussein Suleman	021 6505106	hussein@cs.uct.ac.za
B.2 Co-Supervisor/s			

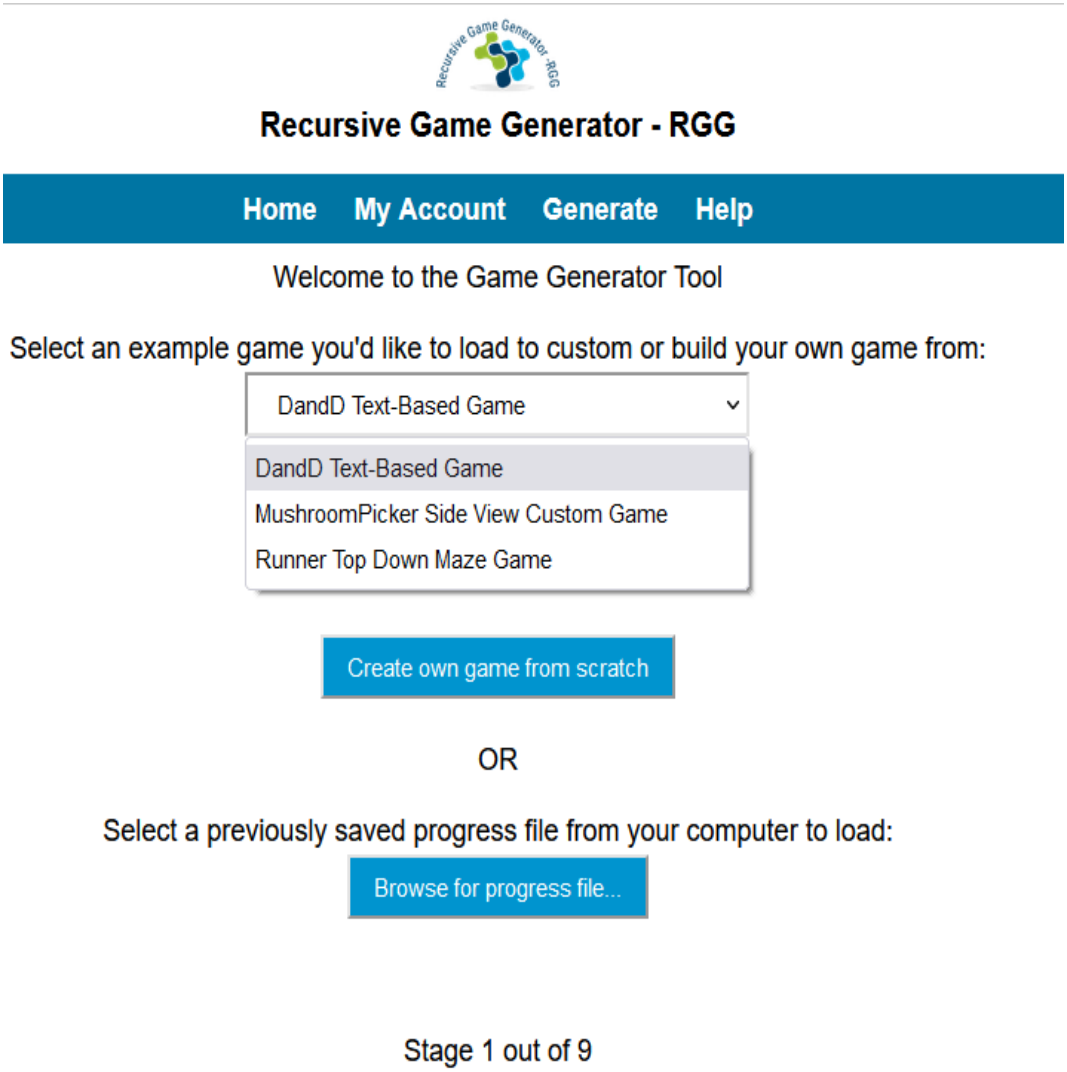
SECTION C: APPLICANT'S RESEARCH STUDY FIELD AND APPROVAL STATUS

C.1 Degree – if applicable	PhD.
C.2 Research Project Title	An Investigation of a Game Generator Tool to Teach Recursion
C.3 Research Proposal	Attached: Yes ■ No ☐
C.4 Target population	Students – Registered Introductory Programming (CS1) students at UCT
C.5 Lead Researcher details	If different from applicant:
C6. Will use research assistant/s	Yes ☐ No ■ If yes- provide a list of names, contact details :
C.7 Research Methodology and Informed consent	**Research methodology:** Quantitative online gaming and questionnaire. **Informed consent:** Advised to participants for consent to articpate.
C.8 Ethics clearance status from UCT's Faculty Ethics in Research Committee /Chair (EiRC)	Approved by the UCT EiRC: Yes ■ With amendments: Yes ☐ No ■ (a) Attach copy of your UCT ethics approval. Attached: Yes ■ No ☐ (b) State date / Ref. No / Faculty of your UCT ethics approval: 16/06/2021 Ref. / Faculty: FSREC 055-2021

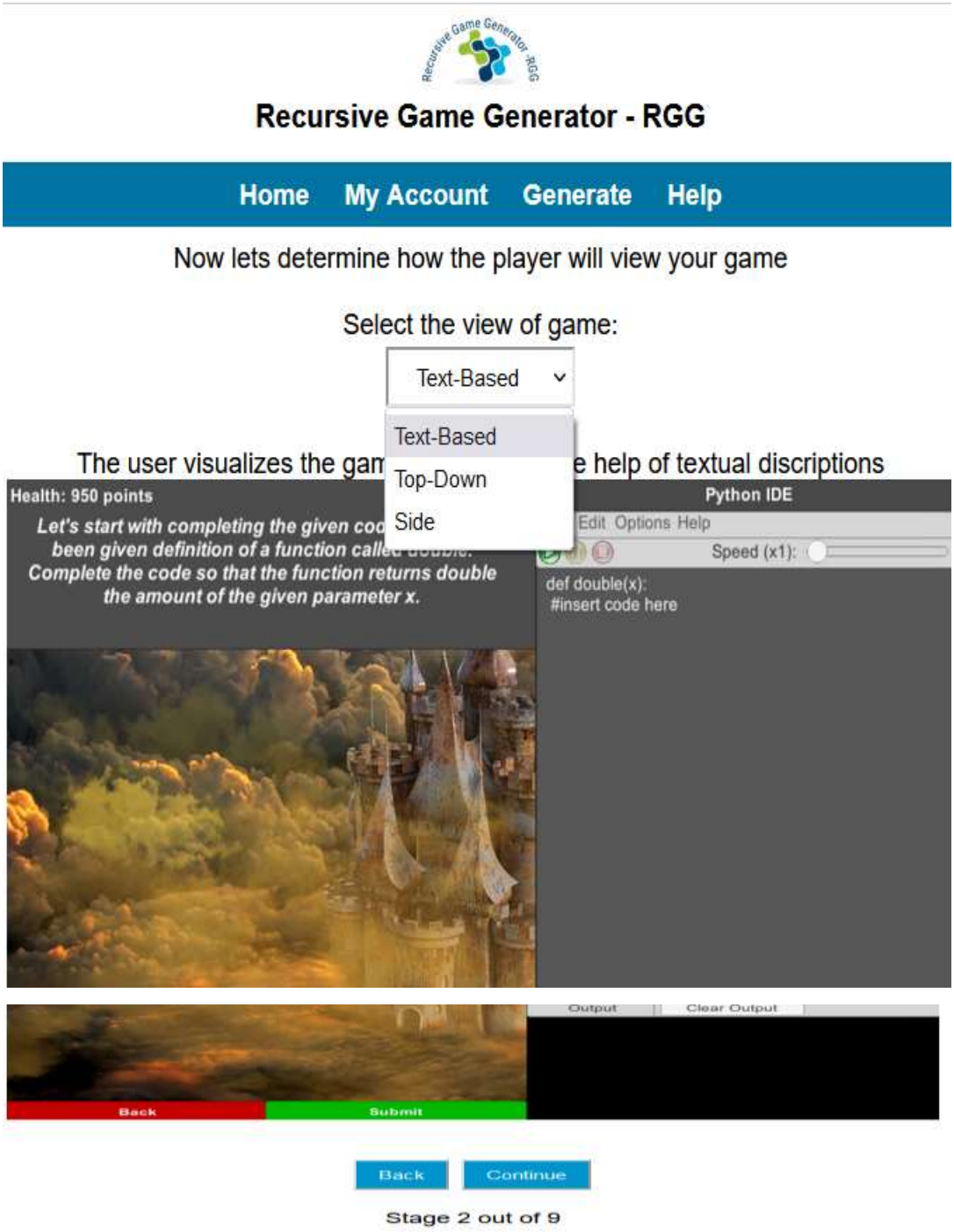
SECTION D: APPLICANT/S APPROVAL STATUS FOR ACCESS TO STUDENTS FOR RESEARCH PURPOSE
(To be completed by the ED, DSA or NOMINEE)

D.1 APPROVAL STATUS	Approved / With Terms / Not	* Conditional approval with terms		Applicant/s Ref. No.:
	(i) Approved ■ ~~(ii) With terms~~ ~~(iii) Not approved~~	a) Access to students for this research study must only be undertaken after written ethics approval has been obtained. *b)* In event any ethics conditions are attached, these must be complied with before access to students.		ANYJEC001 / Mr. Jecton Anyango
D.2 PREPARED BY:	Designation	Name	Signature	Date of Approval
	Personal Assistant	*Nadierah Pienaar*		13/07/2021
D.3 APPROVED BY:	Designation	Name	Signature	Date of Approval
	Executive Director Department of Student Affairs	*Mr Pura Mgolombane*		*14/07/2021*

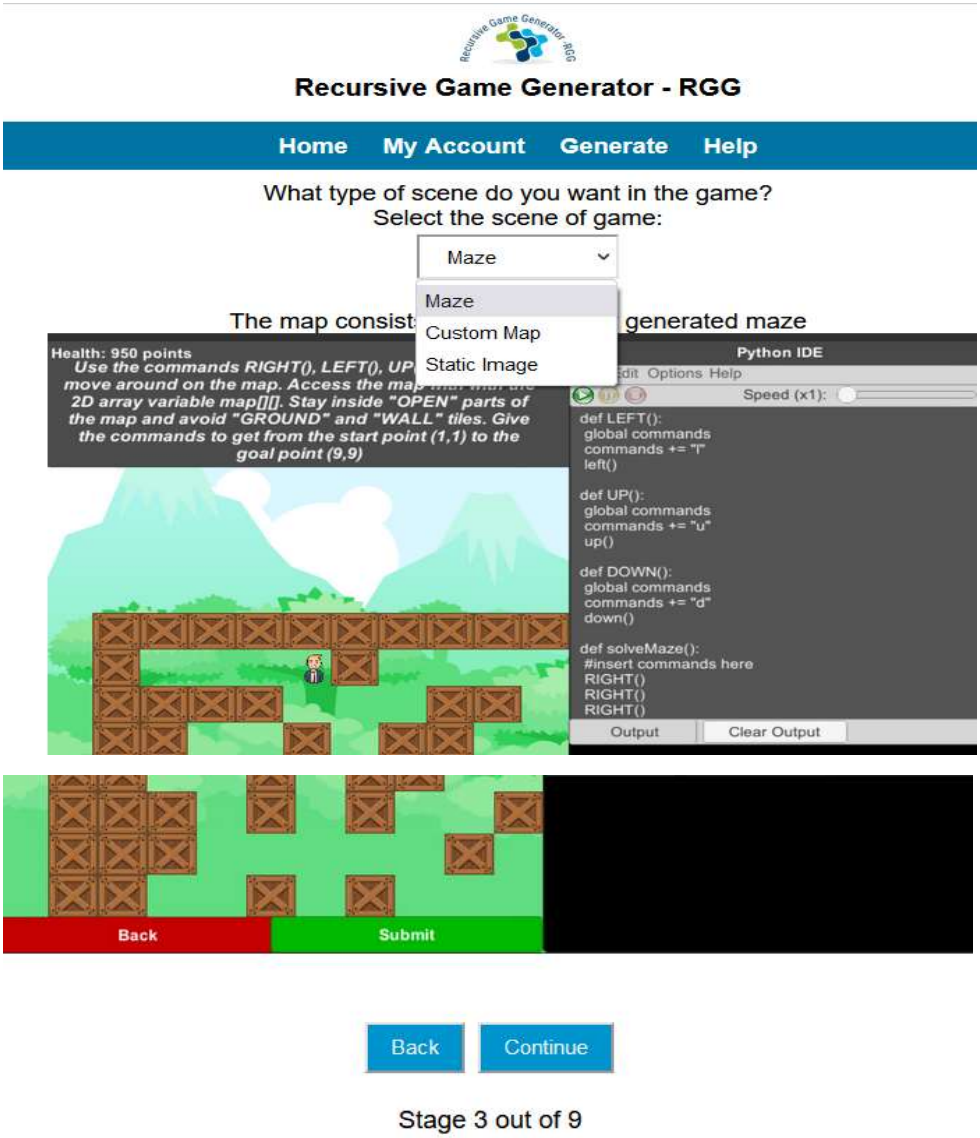
C.21 Game generation step 1



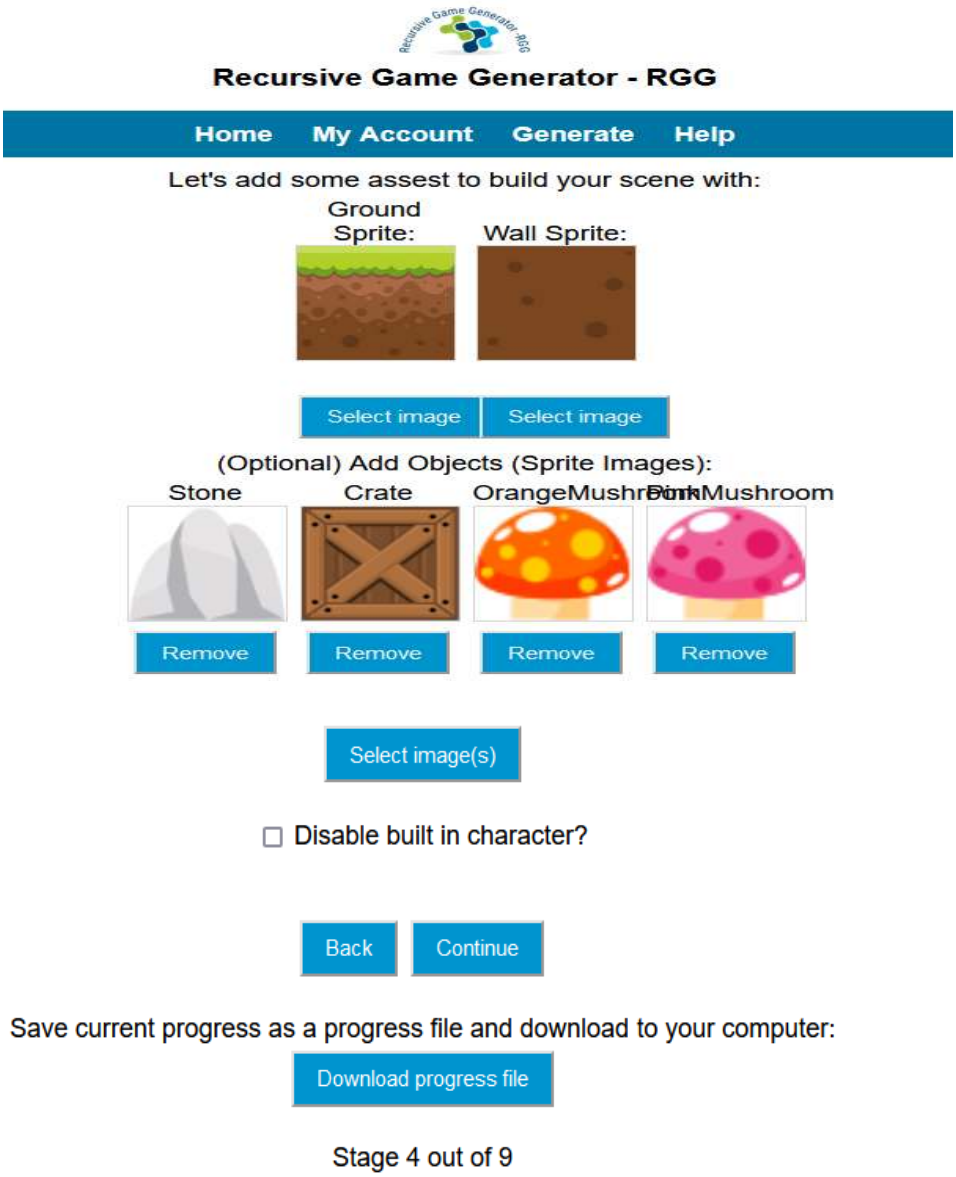
C.22 Game generation step 2



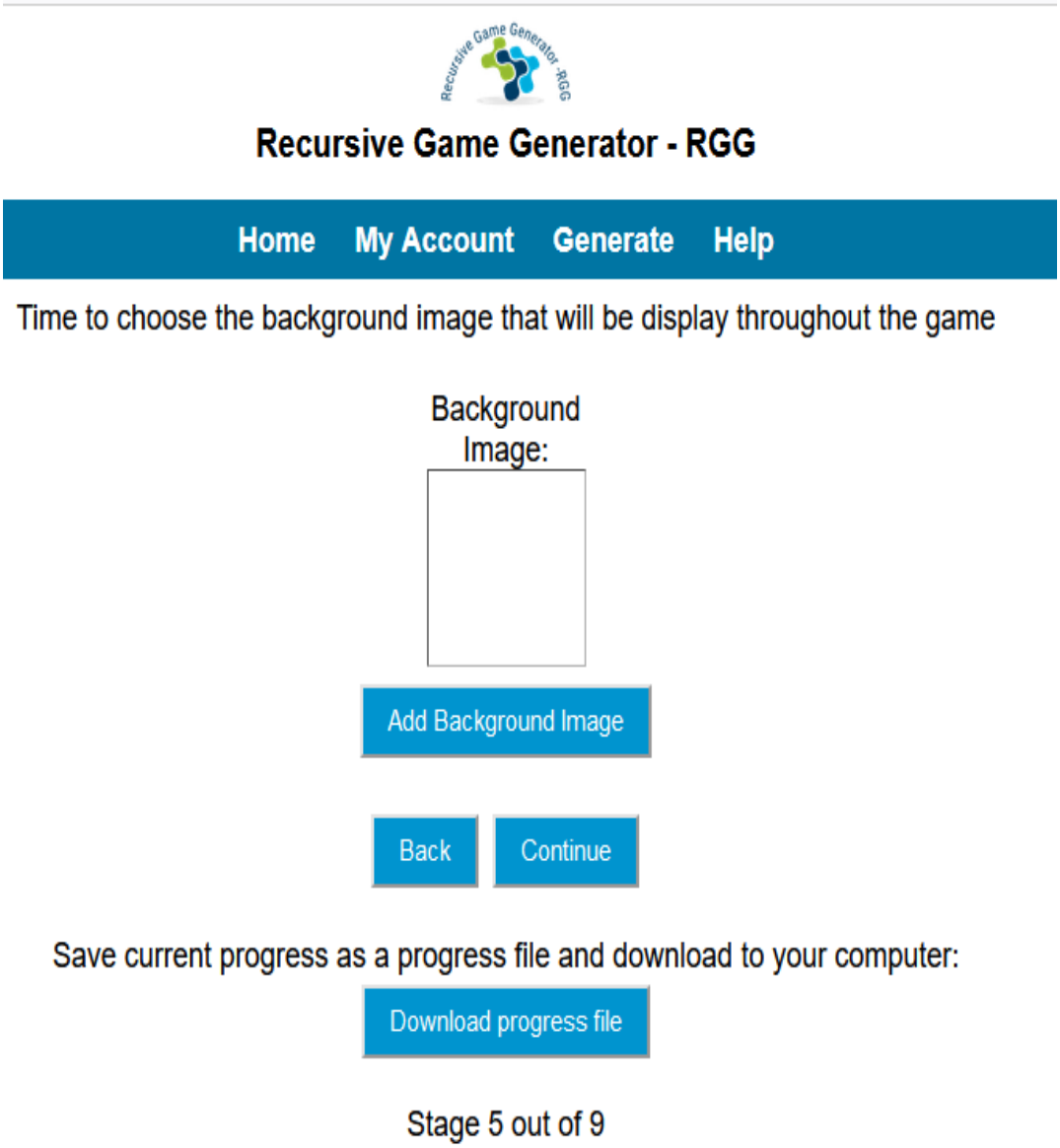
C.23 Game generation step 3



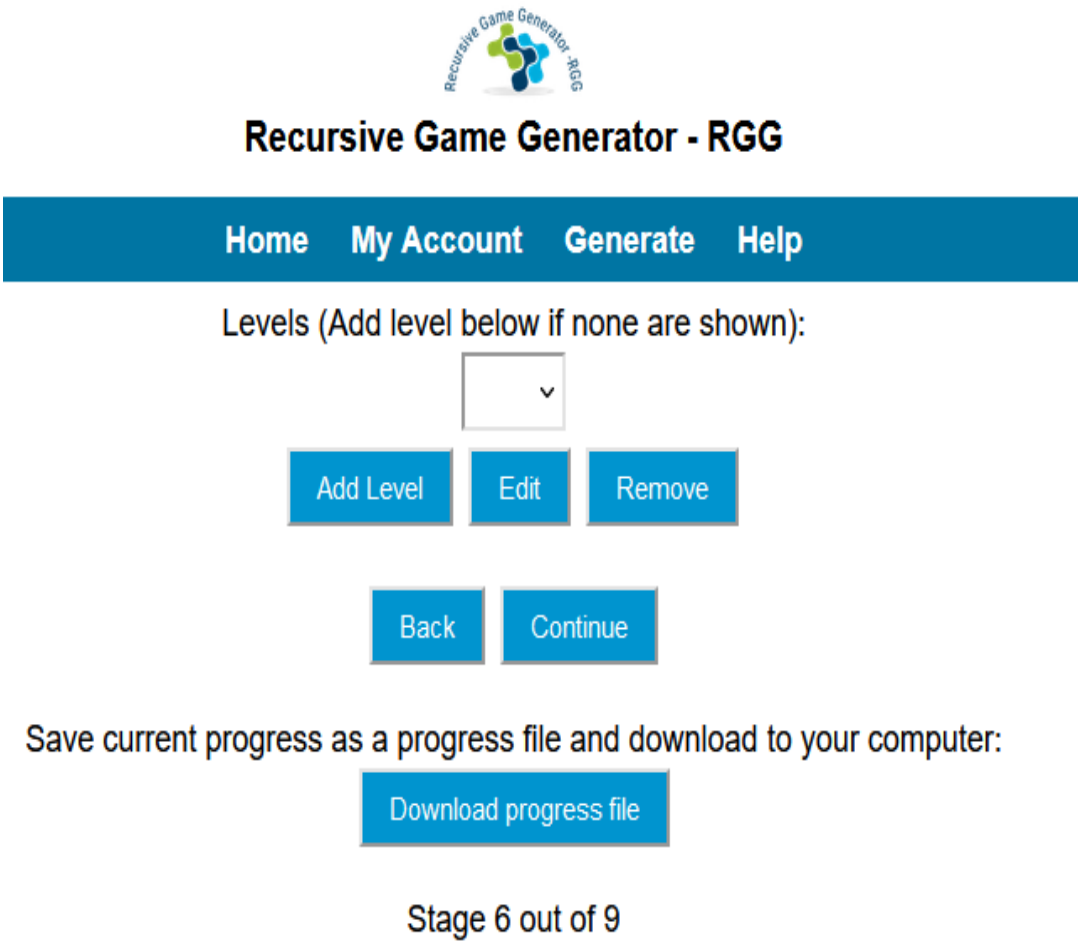
C.24 Game generation step 4



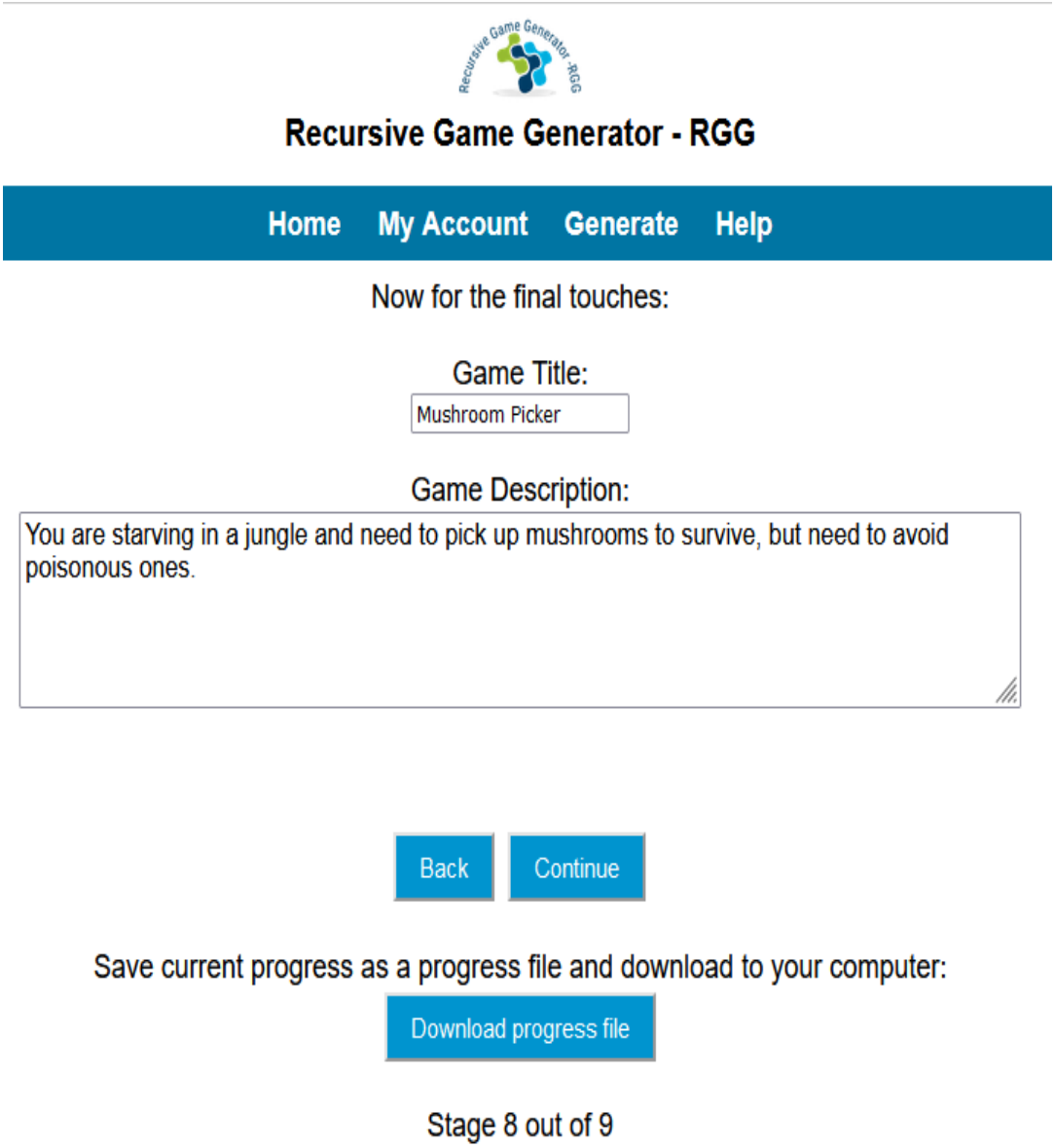
C.25 Game generation step 5



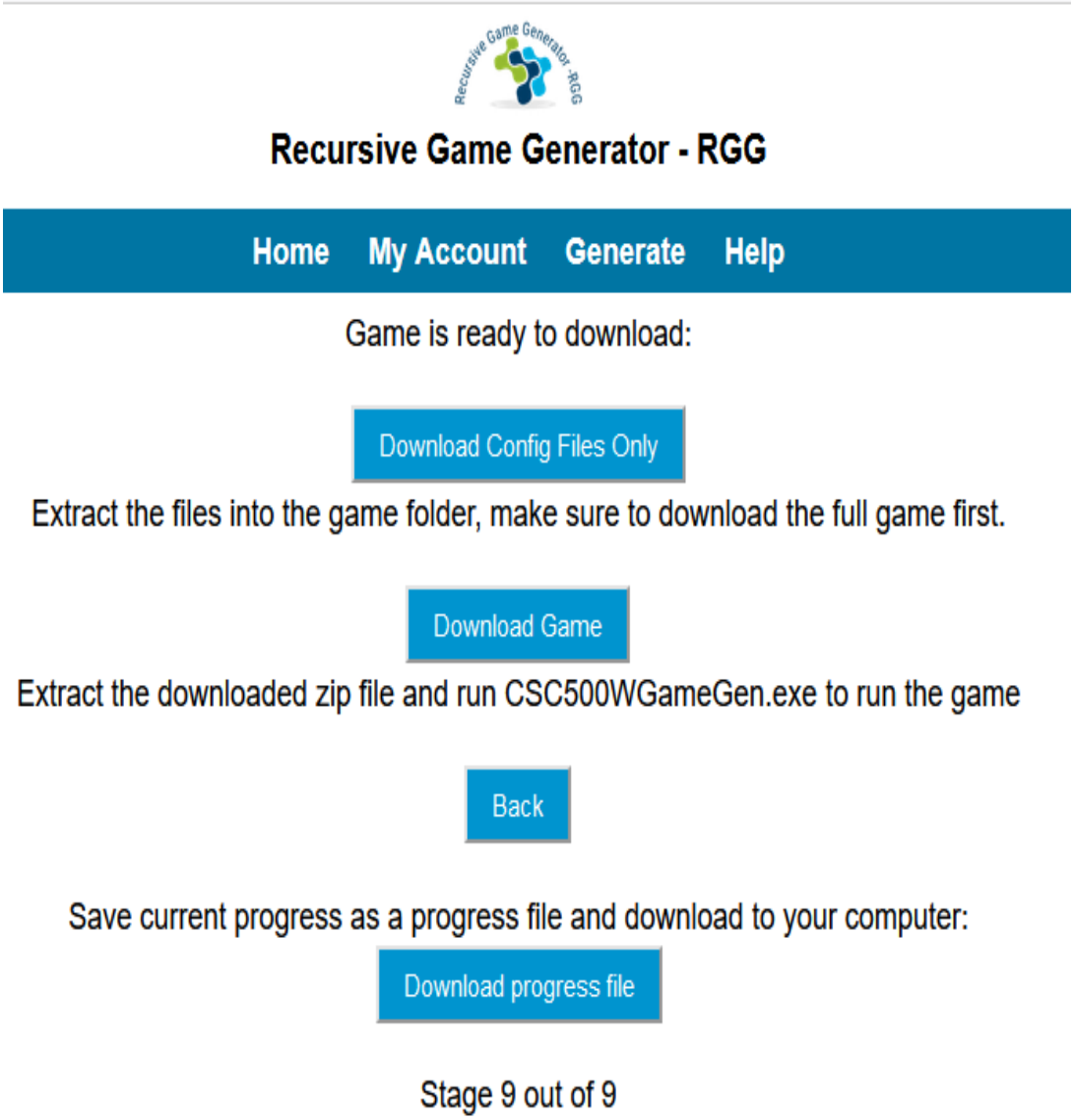
C.26 Game generation step 6



C.27 Game generation step 8



C.28 Game generation step 9



C.29 Trophies and health points as reward