Volcanic Skyscapes

P. Cilliers Pretorius

Supervisor: James Gain

Minor dissertation presented in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE

in the Department of Computer Science,

Science Faculty, University of Cape Town.

22 July 2024

Abstract

Explosive volcanic eruptions are some of the most visually impressive natural phenomena, and are frequently represented in visual media such as films or games, usually with a bespoke, one-off solution. Existing plume models treat the atmosphere as a static boundary, while existing atmosphere models are not designed for the extremes present in a plume. We present the first general-purpose model with bi-directional interaction between a volcano plume and the atmosphere. The atmosphere is modelled as sparse layers of 2D Eulerian grids, while the plume dynamics are modelled with Lagrangian disks and spheres. We enable the transfer of physical quantities such as ash, moisture, temperature, and wind velocity between these submodels. Volumetric animations are generated through procedural upsampling using noise functions, specifically tuned to the physical processes in play, to create a complete volcanic skyscape. Our model depicts most of the relevant facets arising from plume and atmosphere interactions, such as ash rain, embedded bell, cap, and skirt clouds, shockwave effects, and volcanic lightning.

Plagiarism Declaration

I know the meaning of plagiarism and declare that all of the work in the dissertation, save for that which is properly acknowledged, is my own.

Pieter Cilliers Pretorius
22 July 2024

Acknowledgements

I would like to firstly thank my supervisor, Professor James Gain, for his advice, patience, and support throughout the entire process. Without his advice and insight, this work would not be anywhere near its current standard. I would also like to thank the many friends and family who supported me over the many long nights and weekends, serving as soundboards and providing assistance with comprehending the complexities of fluid dynamics, Sané, Leighton, and Alec in particular. This has been a long journey which would not have been completed without the support from too many people to name.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

"The sea seemed to roll back upon itself, driven from its banks by the convulsive motion of the earth. On the other side, a black and dreadful cloud, broken with rapid, zigzag flashes, revealed behind it variously shaped masses of flame: these last were like sheet-lightning, but much larger. Soon, the cloud began to descend and cover the sea." Such were the words of awed horror, written by Pliny the Younger to the Roman historian Tacitus, describing the eruption of Mount Vesuvius in 79 CE.

Volcanic eruptions have captured the human imagination, across history and across cultures. As much as 10% of the world's population live in close proximity to active or recently active volcanoes[SN01]. Volcanoes can dramatically alter the landscape and ecosystems surrounding them, or even affect global weather patterns for months or years. Given this, it is perhaps no surprise that volcanic eruptions are often depicted in digital media, such as films or games. Beyond the visually spectacular lava flows, the volcanic plume is the core component of any scene including an erupting volcano. Visualising the plume is also of significant importance in both public education about volcanoes and risk forecasting, given the population potentially at risk in volcanic eruptions.

However, depicting a plume in digital media is usually done through an expensive, bespoke, one-off solution. The greatest challenge in depicting these plumes lie in the complicated interactions between the plume, the atmosphere, the terrain, and the effusive ejecta, such as lava flows and lahars (a violent slurry of pyroclasts, mud, and debris moving at high speeds). These require very expensive simulations to accurately model, which, while prevalent in the geophysical sciences community, are not suitable for graphical applications.

Existing graphical simulations of the plume[MDN02b; Miz+03; Las+22] focus on the plume dynamics and treat the atmosphere as a static boundary condition. For instance, Lastic et al.[Las+22] assumes a constant wind velocity at a given altitude. While this method can result in visually appealing plumes, it is not physically accurate with regards to the influence of the atmosphere's advective and convective forces on the plume. Similarly, it ignores the effect that the plume has on the atmosphere through the transfer of heat, moisture, and ash.

While graphical simulations of the plume are limited in number, cloud modelling

and animation in computer graphics is a mature and rich field of research. Despite covering a wide variety of cloud types[Vim+20] and energy dynamics[Häd+20], these are not capable of accurately modelling the plume due to accuracy constraints resulting from modelling typical atmospheric conditions. Volcanic eruptions present extremes in temperature at the vent and in the plume, while also extending high above the usual altitudes allowed for by these atmospheric models. They also fail to take any ash content in the atmosphere into account, which is crucial for key plume features such as ash rain. Given the extremely high energy dynamics present in a volcanic eruption, the time steps employed by atmospheric models are usually too large to capture the fine detail arising from extreme turbulence in the plume.

In this dissertation, we focus on specific phenomena arising from the bi-directional interaction of the plume and atmosphere. We focus on explosive eruptions since these eruptions have the most visually striking phenomena. In particular, cold atmospheric air entrained (taken up and mixed into the plume) into the plume can give rise to embedded clouds, while ash, moisture, and ice within the plume lead to lightning strikes. Shockwaves arising from the initial explosion can lead to transient condensation clouds forming if atmospheric conditions are suitable. Ash sedimenting out of the plume and subsequently mixing with moisture in the atmosphere leads to the formation of ash rain that precipitates downwind of the plume. This ash rain often has the most prominent effect on civilisation, disrupting transport and damaging buildings and infrastructure on large scales. At the same time, it greatly enriches the soil, leading to very fertile terrain for agriculture.

We aim to simulate the phenomena described above with a model combining two sub-models. To address the specific challenges that arise from plume and atmosphere interaction, we combine a Lagrangian model of plume dynamics[Las+22] with an atmospheric model using Eulerian advection and procedural convection[Vim+20]. We find it necessary to extend these models to handle extreme temperatures and altitudes, as well as to enable a variety of phenomena that were previously unsupported.

The plume is approximated as a stack of cylinders, with movement determined by Lagrangian dynamics arising from ejection and air entrainment forces, as well as temperature, moisture, and wind velocity provided by the atmospheric model. Lagrangian spheres of various sizes are created on the cylinders to approximate a coarse shape for the plume, while also transferring temperature, moisture, and ash to the atmospheric model.

The atmospheric model is structured as sparse two-dimensional layers, with each layer being a Eulerian grid with pertinent quantities such as temperature, moisture, water, ash, and wind velocity. Horizontal wind is simulated through a semi-Lagrangian advection scheme, while vertical uplift due to convective forces and orographic effects is modelled pseudo-physically for computational efficiency. We utilise packet transport calculated by the moist static energy gradient. The moist static energy is an approximation of the total energy in an air packet, including heat energy, potential energy due to gravity, and kinetic energy. Transport will occur from packets of high energy to packets with lower energy. We calculate a pressure adjustment resulting from interactions with the terrain. Heat, moisture, and ash is

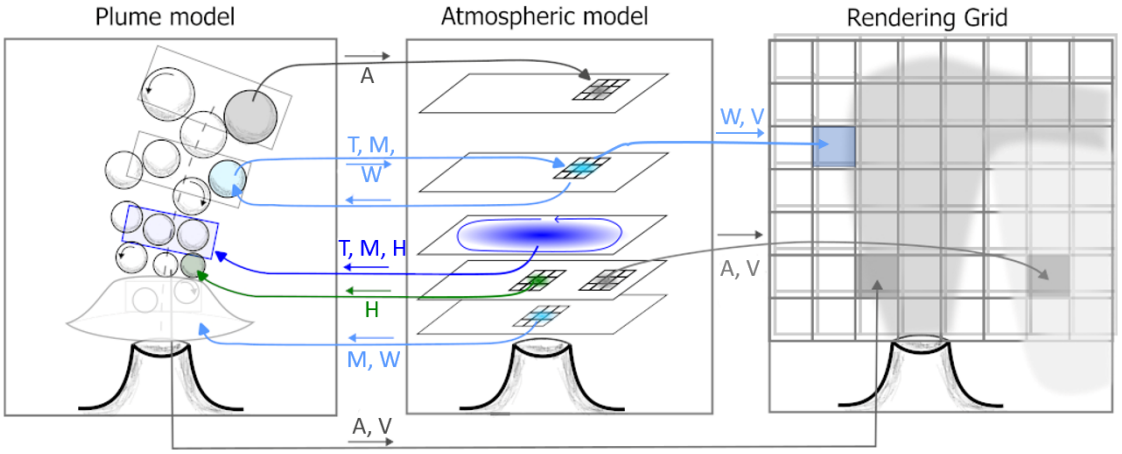


Figure 1.1: We transfer physical quantities between the plume and atmospheric submodels, before outputting the results to the final rendering system. We transfer ash A, moisture density M, water content W, temperature T, 3D velocity V, and horizontal 2D velocity H.

transferred from the plume to the atmospheric model and is used to induce cloud formation, convection, and ash rain once the plume has developed sufficiently.

The output of this coupled simulation is fed to a procedural upsampling process. This results in a finely-sampled three-dimensional grid for the atmosphere, and a collection of ellipsoids with variable density for the plume. This can then be used for offline volumetric rendering, or exported for use as part of a scene in films, games, or educational programs. By utilising an upsampling approach, we generate a plausible, fine-scale animation while only requiring a relatively coarse simulation, resulting in computational efficiency gains. The coarse simulation does result in the loss of some of the complex turbulence, which is then replaced by the output from procedural noise functions. Figure 1.1 provides an overview of the system and the quantities transferred between the atmospheric and plume submodels.

In this dissertation, we present our framework, which couples efficient and interacting sub-models for the volcanic plume and atmosphere, and combines it with procedural amplification. Our model enables the generation of several phenomena previously unavailable in computer graphics, such as ash rain, embedded clouds in the plume, and volcanic lightning. Chapter 2 provides the background for atmospheric physics and plume dynamics that give rise to the phenomena we model. Chapter 3 provides an overview of previous work in this field, as well as related work in other fields. The atmospheric sub-model is described in chapter 4, while the plume dynamics are described in chapter 5. The coupling scheme between the two sub-models and procedural amplification processes are described in chapter 6. We present our results, performance timings, and validation in chapter 7.

Chapter 2

Background

In this chapter, we provide a primer on the movement of air within the atmosphere, its causes and the factors that influence it, and an overview of the water cycle and the resulting cloud formation. We restrict our overview to the troposphere, which contains more than 75% of the atmosphere's total mass and nearly all of its water. We also provide a primer on explosive volcanic eruptions, the dynamics present in the volcanic plume, and factors that influence the appearance of the plume.

2.1 Movement in the atmosphere

Our work is focused on a limited domain, both in space and in time. Explosive volcanic eruptions are comparatively short-lived, rarely lasting more than 24 hours. Similarly, while these eruptions can have effects lasting several years (such as the eruption of Mount Tambora in 1815), the visually salient features rarely last more than a few hours after the eruption comes to an end. Therefore, we will cover only the salient points of weather formation for our scenario. For more detail on weather processes as a whole, we refer the reader to works by Barry et al.[BC09; Bar13] and Pielke et al.[Pie+98]. While the same processes govern movement in all atmospheres regardless of composition, we will restrict this work to Earth-like atmospheres with a water cycle (see section 2.2).

To describe movement in an atmosphere, an understanding of its composition and nature is required. An atmosphere consists of a mixture of gases bound to a planet's surface by the planet's gravity. The atmosphere can carry material other than gases, such as liquid water (water clouds) or particulates (dust or ash clouds).

All movement in the atmosphere (collectively referred to as wind) is a result of a pressure-gradient force. This is the force applied to air molecules as a result of a difference in pressure within a volume, with the force pushing from an area of high pressure to an area of low pressure. As per the general gas equation:

$$pV = nR^*T, \tag{2.1}$$

pressure p is proportional to the amount of substance n and temperature T, and inversely proportional to volume V, where R^* is the universal gas constant. If we discretise the atmosphere, then each parcel or packet of air under consideration will

have a volume that remains constant. Thus, the only differences that can lead to a pressure differential are differences in the temperature T, or in the amount of substance n.

The atmosphere on Earth (at speeds below Mach 1) behaves as an incompressible fluid. Thus, modelling the movement of air is done through the equations for fluid dynamics. It also means that the shape of the underlying surface can have a significant effect on the wind field. In our case, the underlying surface is the terrain. See section 2.4 for a detailed explanation of the fluid dynamics required for our model.

Due to the effect of gravity, the pressure at a given altitude remains relatively constant, with only minor variations. This is expressed by the barometric equation:

$$P_h = P_b\left(1 - \frac{L}{T_b}h\right)^{\frac{gM}{R^*L}}, \tag{2.2}$$

where P_b refers to the base pressure at sea level (which is 101325 pascal on Earth), L refers to the lapse rate, the change in temperature as a function of altitude (6.5 $K \cdot km^{-1}$ on Earth), T is the average temperature at sea level (288.15 K on Earth), and h is the altitude in question. In the exponent, g is the gravitational acceleration (9.8 $m \cdot s^{-2}$ on Earth), M is the molar mass of air (0.029 $kg \cdot \text{mole}^{-1}$ on Earth), R^* is the universal gas constant, and L is once again the lapse rate. However, there are processes that can create sufficiently steep horizontal pressure gradients for weather phenomena to arise despite the effects of gravity.

The density of an air parcel can be approximated as $\rho = n/V$, following the general gas equation (equation 2.1). It follows that an air parcel with a higher temperature will be less dense compared to a parcel with a lower temperature. Since the atmosphere is in the planet's gravitational field, buoyancy forces act on parcels of differing densities. Therefore, warm air will tend to rise above cold air.

The majority of an atmosphere's energy budget comes from the photons emitted by a star. Depending on the wavelength of the photons and atmospheric composition, not all of the energy is directly absorbed by the atmosphere. In the case of Earth's atmosphere, the vast majority of the photons are in wavelengths that the atmosphere is transparent to. Some of the photons are reflected back out to space (depending on the albedo of the surface). The rest of the photons are absorbed by the planet surface, where the photon's energy adds to the temperature of the surface.

The surface radiates this energy as greybody radiation peaking in the infrared, and the atmosphere is opaque to infrared wavelengths. Thus, some of the radiated energy is absorbed by the lower layers of the atmosphere. Some energy is also exchanged through direct contact and conductance from the surface and the air touching it, but since air is a terrible conductor and very low density, this conductance is negligible compared to the radiative heat transfer.

As a consequence of the heating mechanisms of the atmosphere, warm air will form at the surface and then rise due to buoyancy. Such a convective column will

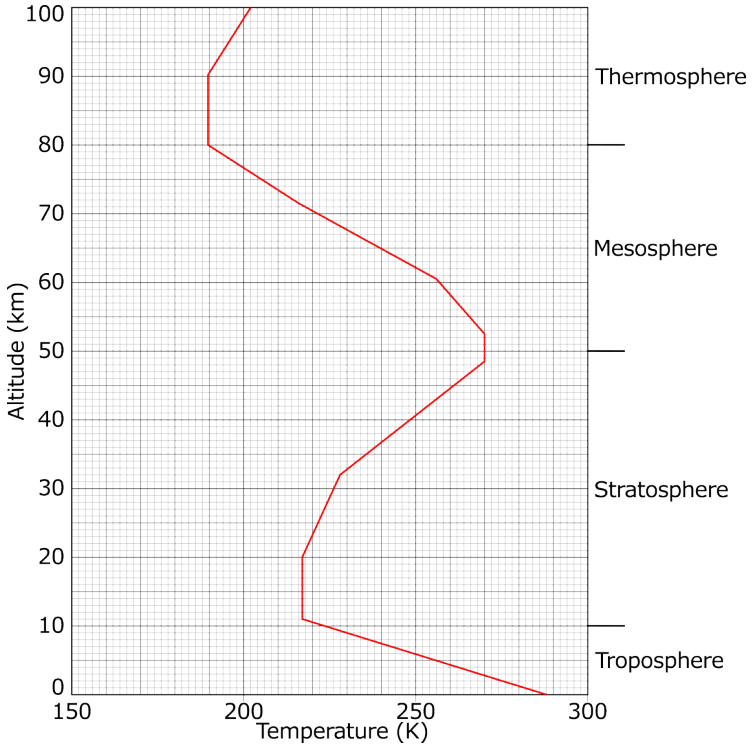


Figure 2.1: Average atmospheric temperature as a function of altitude, with temperature on the x-axis and altitude on the y-axis[OA76]. Note the lapse rate tending to 0 between the altitudes of 10 and 20 km.

tend to have a low pressure zone at its base relative to the average pressure at that altitude. Similarly, the top of the convective column will tend to have a high pressure zone relative to the average pressure at that altitude. This convective cycle creates pressure gradients in the horizontal and thereby drives wind currents.

Another facet of convective columns to take into account is the *tropopause*. As a hot air parcel rises, it exchanges some of its thermal energy for potential energy, as a result of adiabatic cooling. This is cooling as a result of a drop in pressure in the parcel as it rises. As per equation 2.1, if pressure reduces, but volume and amount of substance remain constant, then temperature must reduce as well. At some point, the air parcel temperature matches the temperature of the surrounding air, and therefore reaches the same density. At this point, buoyancy effects no longer apply and the air parcel will not rise any further.

The average lapse rate in the atmosphere is sharply negative for the lower parts of the atmosphere, averaging around 6.5 K per kilometre of altitude. This changes at around 11 kilometres of altitude, where the lapse rate suddenly transitions to zero. See figure 2.1. This boundary is known as the tropopause, and weather phenomena rarely cross it. Explosive volcanic eruptions and the subsequent plumes are one of the few natural phenomena that can penetrate the stratosphere (the layer above the tropopause). Very energetic convective columns that produce cumulonimbus clouds can, in exceptional cases, do this as well. Barring those exceptions, convective columns will tend to terminate near the tropopause, restricting surface weather effects to the lower 11 kilometres of the atmosphere.

2.2 Cloud formation

Water clouds (often referred to as simply clouds) are collections of liquid water (droplets) or solid ice suspended in the atmosphere (technically referred to as an *aerosol*) in sufficient density that the majority of light passing through the area is scattered such that the collection becomes visible. Clouds can consist of material other than water, such as other liquids like carbon dioxide or methane or particulates like dust or ash. In this work, we only consider clouds consisting of liquid water and ash.

Water vapour (gaseous H_2O) makes up a small, but highly variable portion of the Earth's atmosphere[RLB19]. Typically, it makes up around 1% of the atmosphere at sea level, though it varies from negligible to 3% of the atmosphere. Higher altitudes and latitudes will generally have lower concentrations of water vapour, since the amount of water vapour that a given parcel of air can hold is dependent on the temperature of the air. Higher temperatures mean a higher capacity for water vapour. Some of the water vapour will condense out into its liquid state. Similarly, if there is liquid water in the atmosphere, some of the liquid water may evaporate to its gaseous state.

The rate at which the two states of water change depends on the *saturation vapour pressure* for the parcel. This is the point where the quantities of liquid water and gaseous water vapour are such that there are equal numbers of molecules changing to and from each state. Specifically, the water vapour will tend to be in thermodynamic equilibrium with the liquid state. The saturation vapour pressure increases with temperature, which is why air at a higher temperature can hold more water vapour. For reference, the saturation vapour pressure is equal to one bar (101325 Pascal) at a temperature of 100 degrees Celsius. See section 4.2.6 for a more detailed look at this phenomenon and its implementation in this work.

2.2.1 Water cycle

Water vapour (hereafter referred to as moisture) enters the atmosphere by evaporating from the Earth's surface in some fashion. This can be from open water expanses like oceans, lakes, and rivers, or it can be from moist soil that dries out, or from transpiration from plants. In this work, we focus only on the abiotic sources, though for a fully realistic simulation, the biotic sources and influences cannot be ignored.

Once the moisture is in the atmosphere, it changes state to remain in thermodynamic equilibrium depending on the temperature of the air. Air is heated largely through infrared radiation from the ground, and is cooled largely through radiation to space, conducting to colder surfaces (such as land at night), and adiabatic cooling, that is, cooling due to a drop in pressure (see equation 2.1). This pressure drop is a result of the air parcel rising higher, whether because of convection or being forced up by a collision with the terrain (known as orographic uplift). The air molecules exchange kinetic energy, i.e., heat, for potential energy in the gravitational field. If moisture and liquid water changes state, that also has an effect on the temperature, but it is less influential than adiabatic cooling. See section 4.2.6 for an examination

of the temperature changes that result from changing states.

Moisture is indirectly removed from the atmosphere and returned to the surface through precipitation which removes liquid water from the atmosphere, and thus moisture will condense out to liquid water to restore the thermodynamic equilibrium. Moisture cannot condense out without some sort of condensation nucleus. This can be ice, dust, or other particles. In the atmosphere of the Earth, it is rare for conditions to exist where condensation nuclei are not present in sufficient quantities to satisfy all condensation, and we do not consider those circumstances in this work.

2.2.2 Precipitation

Precipitation refers to the various forms of non-gaseous water that fall from clouds due to the gravitational attraction of the Earth. Precipitation can be solid (hail or snow), or it can be liquid (rain or drizzle). For this work, we consider only liquid and ash precipitation. Liquid precipitation makes up the vast majority of precipitation, while ash precipitation is of particular visual interest in the case of volcanoes.

Liquid precipitation

Rain falls when some of the water droplets that make up a cloud grow too large to be kept in the air through electrostatic forces or air resistance. Droplets grow through *coalescence*, which is where two or more droplets combine to form a larger droplet. This is a result of collision, and turbulent conditions within the cloud lead to faster coalescence[Voh+99]. Once a droplet has grown large enough to overcome air resistance, it starts falling. Depending on the conditions below the droplet, it might continue to grow through collision with other droplets (such as in strong convective clouds), or it might evaporate before it hits the surface (known as virga). Once the rain has reached the surface, it can be taken up by open water, or it can be absorbed by the soil (turning the soil moist). It thus becomes available for uptake into the atmosphere again, starting the water cycle afresh.

Ash precipitation

Ash rain is a common sight in the vicinity of volcanoes undergoing an eruption. It is formed when fine ash particles mix with liquid water in the atmosphere, or serve as condensation seeds for water to condense around. These particles of mixed water and ash will grow through collision similar to the coalescence of liquid precipitation, though it is called *aggregation* in the case of ash rain[BBD12; RD11]. Many factors influence the size and granularity of the ash particles in a volcanic plume[Van+15], such as the chemical composition of the volcanic magma, the geology of the terrain, and the explosivity of the eruption, but we do not consider those intricacies in this work.

This ash rain can pose a hazard to people in the vicinity of an eruption, often smothering existing plant life, and can suffocate animal life if the ash fall is heavy. It also presents a significant logistical challenge when clogging roads and railways, or when it has to be removed from buildings[Jen+15]. However, this ash also greatly enriches the soil, leading to highly fertile ground surrounding volcanoes.

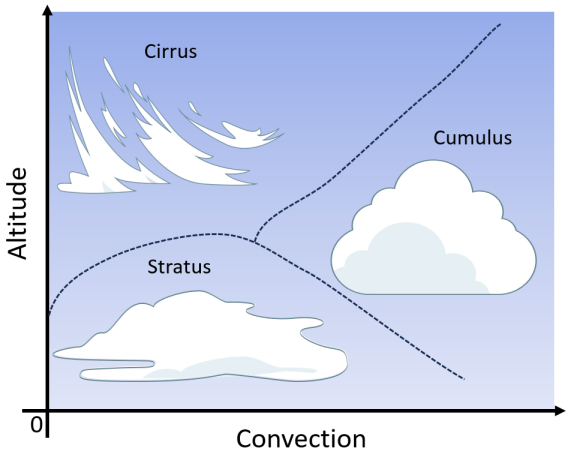


Figure 2.2: Cloud classification according to height and convective activity, with activity increasing to the right.

2.2.3 Cloud types

Clouds are classified into various different types according to the altitude (or level) of the clouds (which generally dictate the composition, whether majority liquid or majority ice), as well as the convective structure within the cloud (which dictates the shape of the cloud). The level of a cloud is generally indicated with a prefix, with no prefix for low level clouds, *alto-* for mid level clouds, and *cirro-* for high level clouds. Multilevel clouds usually get their own designation depending on the convective type.

In this work, we consider only a simplified cloud classification scheme, as shown in figure 2.2. We refer the reader to the International Cloud Atlas[Org17] for specifics on cloud types, as well as more information on cloud classification in general.

Cumulus clouds are clouds with a high degree of convection, characterised by a "fluffy" appearance. These clouds are almost entirely liquid, leading to the fine edge due to the strong electrostatic bonds between liquid water molecules. *Cirrus* clouds are wispy, filamentary clouds, with little convective activity, and occur only at high altitudes (and therefore do not get a prefix). These generally consist of ice crystals, which have weaker electrostatic bonds compared to liquid water, and this results in less defined edges compared to other types of clouds. *Stratus* clouds are flat sheets of cloud, with little to no convective activity.

2.3 Explosive volcanic eruptions

A short overview of explosive volcanic eruptions is required for an understanding of our model. In this work, we focus on the evolution of the volcanic plume and place less emphasis on the terrain altering effects, such as the formation of calderas or magama chambers, or the presence of lava or lahars. For a more complete treatment of volcanic eruptions, we refer the reader to works by Cas and Wright[CW12] and Parfitt and Wilson[PW09].

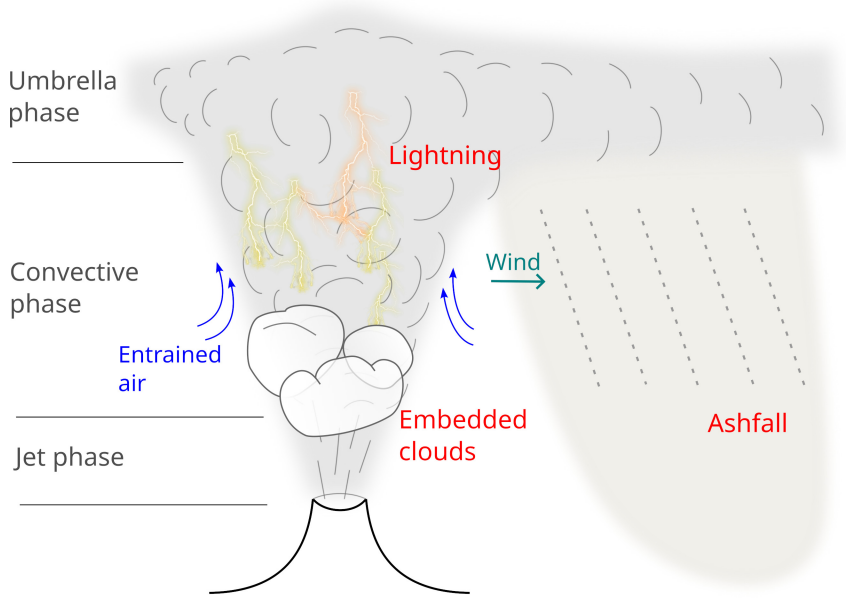


Figure 2.3: The typical progression of a Plinian eruption, divided into the jet, convective, and umbrella phases. The salient visual features resulting from an eruption and the phases where they occur are indicated as well.

There are many types of volcanic eruptions. We focus on a subset of magmatic eruptions, i.e., eruptions that involve the expulsion of molten rock and ash. In particular, we focus on the so-called Plinian or ultra-Plinian eruptions, since these eruptions have the most visually striking interactions with the atmosphere. This is compared to Peléan, Vulcanian, Strombolian, or Hawaiian eruptions. Magmatic eruptions are categorised according to the explosivity of the eruption, with Plinian eruptions being the most explosive and Hawaiian eruptions the least explosive.

Plinian eruptions are caused by excessive pressure within the volcano's magma chamber, often exacerbated by a plug, usually consisting of solidified lava, sealing the volcano vent. Pressure builds as solubles such as water vapour and other gases separate out of the magma, or as additional rock in the chamber is melted and expands. Eventually, the plug gives way and the pressure is released catastrophically. This leads to an eruptive column of gas, lava, and tephra (the fragmented, rocky material erupted by the volcano) travelling at high velocities (50 to 200 $m \cdot s^{-}1$). This column is typically very dense compared to the air, often hundreds of kilograms per cubic metre, despite being extremely hot (in excess of 1200 K).

A Plinian eruption typically consists of three phases, demarcated by the primary forces that drive the movement of the eruptive column. In the initial *jet* phase, the column is lifted by the momentum from the explosion. Once expelled from the volcano vent, the column starts to expand due to the immense heat it has. The densest components of the jet, such as lava, volcanic bombs, and coarse ash, sediments out, further reducing the density of the eruptive column. Additionally, air surrounding the jet is dragged in and entrained in the plume, where it heats up. As described in section 2.1, hot air will expand and rise up due to convective forces. This is the *convective* phase of the eruption. If the initial explosion is not powerful enough for

entrainment to occur, or if the initial jet has a significant horizontal component, the plume can collapse and spread across the terrain. This is typically considered a Peléan eruption.

At some altitude, the volcanic plume will reach a neutral buoyancy, and enter the *umbrella* phase. Depending on the initial force and temperature of the eruption, this might be at an extremely high altitude, up to 58 kilometres above sea level[PPS22]. This is well past the tropopause (see section 2.1), and is the reason that the sulphur and other particulates ejected by volcanic eruptions can have global long-term effects, such as a volcanic winter (a sustained period of lower temperatures across the globe – due to the ash reflecting and blocking sunlight – lasting months to years). Figure 2.3 displays the different phases of the eruption and some of the visual features that occur in the plume and atmospheric interaction.

2.3.1 Volcanic weather phenomena

Each phase of the eruption has distinct weather phenomena associated with it. We describe some of the most visually striking examples, which we focus on in this work, and provide reference images.

Transient condensation clouds

In the initial explosion, the rarefaction phase of the shockwave causes a drop in pressure and temperature. In humid atmospheres, this can lead to a very short-lived imprint of condensation as the colder air cannot sustain the previous levels of moisture[YI07]. This imprint showcases the composition of the atmosphere around the volcano, highlighting sections of the atmosphere that are more humid. The atmosphere rapidly returns to normal pressures and temperatures, leading to the condensation cloud evaporating again. This phenomenon rarely lasts more than a handful of seconds.

This phenomenon is seldom captured in relation to volcanoes, mainly due to the remote location of many volcanoes and the lack of visual monitoring equipment in close proximity to volcanic explosions. However, it is often captured with man-made explosions, such as munitions tests or accidents like the 2020 Beirut explosion. Figure 2.4 shows a TNT explosion with an associated transient condensation cloud.

Figure 2.5 shows two transient condensation clouds forming after an explosion of Mount Tavurvur in Papau New Guinea. While this explosion is of a Strombolian nature, Plinian or ultra-Plinian eruptions will be more explosive and have stronger shockwaves. The first transient condensation cloud forms at $t = 0.5\ s$ (figure 2.5(4)) around the plume and dissipates by $t = 1.5\ s$ (figure 2.5(10)). The second, more distant cloud forms at $t = 2.83\ s$ (figure 2.5(14)) in the air near the top of the frame, and disappears from the camera view by $t = 4.5\ s$ (figure 2.5(24)).

Embedded clouds

Along with air being entrained into the plume, air layers can also be forced upwards by the rising plume. Eventually, these air layers mix with the plume. As with

Figure 2.4: A photo of 500 tons of TNT exploding shows the shockwave progress visible on the water and the condensation cloud forming behind it in the air. *Operation Sailor Hat, Hawaii, 1965. Courtesy of the US Navy.*

normal convective columns, the air in these layers cools as it rises. If it rises high enough (or has sufficient moisture), the moisture in the layers can condense to form clouds. Depending on the structure of the atmosphere at the time of the explosion, these layers can take the form of layered "bell" or "skirt" clouds that appear to hang from the plume[Bar82]. Figure 2.6 shows examples of this. It can also form saucer-like pileus or "cap" clouds on top of the plume, as happens occasionally with strong convective clouds. Figure 2.7 demonstrates an excellent example of a cap cloud forming and the plume breaking through it.

Ash rain

While the denser components of the eruptive column will fall out soon after the eruption, the less dense, finer ash can be carried up with the plume, often for dozens or hundreds of kilometres. In the umbrella phase, this fine ash can mix with water droplets to form ash rain. This is a common feature of most volcanic eruptions, though the sheer volume of ash erupted by Plinian eruptions make the subsequent ash rain more striking and long-lasting. Figure 2.8 shows a plume advected by the wind in one direction at its top, but the ash rain is subject to different wind conditions and therefore follows a different direction.

Lightning

The strong convection within a volcanic plume results in friction between particles of ash, ice, and tephra. This friction often leads to electric discharges in and around the plume as volcanic lightning. While singular strikes are brief, many strikes can occur in a short span of time. In addition, these strikes are often spatially extensive, taking the appearance of a web or a sheath enclosing the plume. Figure 2.9 gives an example of volcanic lightning in nature, with the glow from the lava adding a

Figure 2.5: A timelapse showing two condensation clouds forming near the vent of a volcano following an explosion. The first occurs at $t = 0.5$ s around the plume, while a second forms at $t = 2.83$ s in the air near the top of the frame. The full video is available on YouTube. *Mount Tavurvur, Papau New Guinea, 2014. © Phil McNamara.*

Figure 2.6: Skirt clouds are formed by the strong convective uplift around the plume. *Eruption of Hunga Tonga-Hunga Ha'apai, Tonga, 2022. Courtesy of the Tonga Geological Services.*

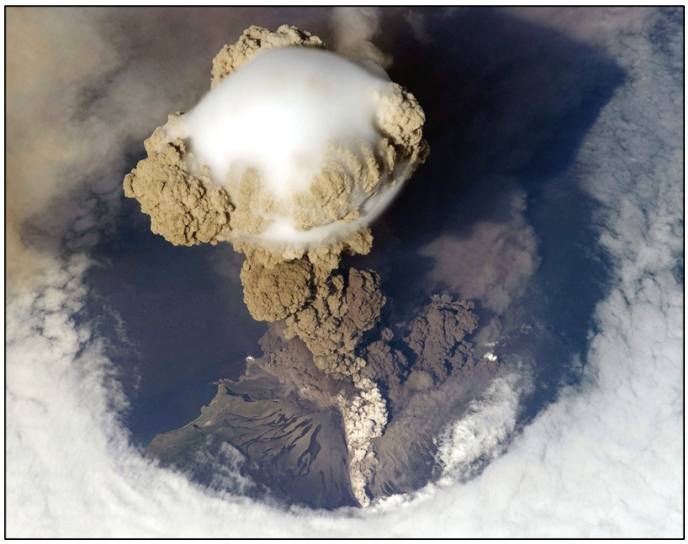


Figure 2.7: A pileus cloud can form at the peak of the plume, before the plume breaks through the cap cloud. *Eruption of Sarychev Peak, Russia, 2009. Courtesy of NASA.*

Figure 2.8: Ash carried up by the plume mixes with water droplets in the upper atmosphere, and then falls to the ground as ash rain. During the fall, it is subject to the local wind field that will dictate the dispersal. *Eruption of Shinmoedake, Japan, 2009. Courtesy of the U.S. Geological Survey.*

striking contrast to the plume.

2.4 Fluid dynamics of the atmosphere

At the heart of our simulation lies the fluid dynamics engines. These are responsible for transport within the layer and ensuring that the wind field remains consistent with the properties of an incompressible fluid. In this section we explain why the atmosphere can be treated as an incompressible fluid, provide the mathematical grounding on the Navier-Stokes equations for incompressible fluids, and go over the implementation of these continuous functions on a 2D discretised grid (see section 4.1 for more details on our implementation).

2.4.1 Dynamics of incompressible fluids

The atmosphere behaves as a fluid, and whether or not a fluid is considered incompressible depends on whether the density changes as a result of the fluid moving through a flow field. This depends on the elastic modulus and density of the fluid, which can be approximated using the speed of sound within the fluid. A higher elastic modulus (i.e., a stiffer material) will have a higher speed of sound, while a higher density will lead to a lower speed of sound. All fluids are compressible under certain circumstances, e.g., high speed movement or confined spaces. The atmosphere is an open system and thus we only have to concern ourselves with the speed at which the fluid is moving.

If the fluid is moving much slower than the speed of sound (the rate at which vibrations within the fluid is transmitted), then any thrust applied to the fluid has time to disperse to neighbouring parcels, and any compression will be negligible. If the fluid is moving closer to the speed of sound within the fluid, then the applied thrust does not have time to disperse, and the parcels within the fluid will be forced closer together, i.e., be compressed.

Figure 2.9: A sheath of lightning surrounding the plume, with a long night time exposure highlighting the multiple strikes. *Eruption of Eyjafjallajökull, Iceland, 2010. ©Terje Sørgjerd.*

The speed of sound within air at standard temperature and pressure is 343 $m \cdot s^{-1}$. Since the temperature (and thus density) of the air varies with altitude, the speed of sound also varies with altitude. The speed of sound will generally decrease with altitude, reducing to about 300 $m \cdot s^{-1}$ at the tropopause[BHR07]. In comparison, the fastest wind speed ever recorded on Earth was 113.3 $m \cdot s^{-1}$ in the eye wall of the category 4 tropical cyclone Olivia in 1996[Cou+12], while the average wind speed across the globe is merely 4.9 $m \cdot s^{-1}$[AJ05]. Even the most extreme case we can expect is well below the speed of sound for air.

While the pressure waves produced by volcanic eruptions can approach the speed of sound[Wri+22], the wind speed resulting will be less given the short-lived nature of the pressure waves. Wind resulting from displaced terrain could potentially be faster, but we do not model terrain displacement and can therefore ignore it. Thus, we can safely model the atmosphere as an incompressible fluid.

2.4.2 Navier-Stokes

The Navier-Stokes equations for incompressible fluids are given by the following equations:

$$\frac{\partial \vec{U}}{\partial t} + \vec{U} \cdot \nabla \vec{U} + \frac{1}{\rho} \nabla p = \vec{g} + \nu \nabla \cdot \nabla \vec{U} \tag{2.3}$$

and

$$\nabla \cdot \vec{U} = 0 \tag{2.4}$$

Equation 2.3 describes the change in the velocity vector field ($\vec{U}$), and is derived from Newton's Second law of motion, $\vec{F} = m\vec{a}$. Equation 2.4 describes the conditions that have to be satisfied if the fluid is incompressible, which is derived from calculating the change in volume with the Divergence Theorem, and having the constraint that $\frac{dV}{dt} = 0$ for all t in our system (where V is the volume of the entire system, i.e., the total volume remains unchanged). See table 2.1 for an explanation of the symbols used in these equations and their units. For the details of the derivations, we refer the reader to *Fluid Simulation for Computer Graphics* by Robert Bridson [Bri15], on which much of this section's work was based.

Symbol	Description	Unit
$\vec{U}$	2D vector velocity	$m \cdot s^{-1}$
t	time	s
ρ	fluid density	$kg \cdot m^{-3}$
p	(modified) pressure	Pa
$\vec{g}$	Vector sum of body forces	N
ν	kinematic viscosity	$Pa \cdot s$

Table 2.1: Symbols and their units used in equations 2.3 and 2.4

In equation 2.3, the acceleration for a given fluid parcel is calculated with three forces affecting it: the vector sum of body forces $-\vec{g}$ (i.e., forces such as gravity); the viscosity $-\nu \nabla \cdot \nabla \vec{U}$ (the fluid's resistance to deformation during flow); and most

importantly, the force exerted by the pressure gradient $\frac{1}{\rho}\nabla p$. This last term is the corrective term that ensures that our velocity field remains non-divergent and satisfies equation 2.4.

We model the atmosphere as a closed system of 2D layers. The only body force applicable to our system is gravity, since we solve the transport equations separately for each layer, neglecting the forces that can arise from vertical transport. These effects are considered in a procedural manner, described in section 4.3.2. We model only the horizontal velocity in our fluid simulation, that is, the velocity parallel to a flat terrain. The acceleration due to gravity is perpendicular to this velocity, and therefore the force as a result of gravity will always be zero in our 2D layers. Therefore, we can discard all body forces in our calculation. The kinematic viscosity ν of air is very low, with magnitudes on the order of 10^{-5}. Therefore, we can discard the viscosity term, though it can cause errors. Interestingly, the numerical diffusion that arises from discretising equation 2.3 will often have similar results to including the viscosity term [Bri15]. These factors combine to give us the simplified equation below:

$$\frac{\partial \vec{U}}{\partial t} + \vec{U} \cdot \nabla \vec{U} + \frac{1}{\rho}\nabla p = 0 \tag{2.5}$$

This, combined with equation 2.4 which still holds, is the final equation we use for modelling movement within layers.

2.4.3 Advection

For transporting the quantities in the layer accurately, we need to solve the **advection** equation, equation 2.6. The quantities need to be moved without changing their values, that is, the quantities need to be *advected*. For a quantity q, the following constraints hold:

$$\begin{aligned} \frac{\partial q}{\partial t} + \vec{U} \cdot \nabla q &= 0 \\ \frac{Dq}{Dt} &= 0 \end{aligned} \tag{2.6}$$

There are two approaches to solving the advection equation, which also tracks and models the movement within a continuous field such as a fluid. See figure 2.10 for a comparison between the two methods. *Lagrangian* methods are particle-based, with each particle having its own position, velocity, and scalar values (see table 4.6). At each timestep, each particle calculates its acceleration as per equation 2.3 and updates its position. This method is conceptually simple and has little or no issue with resolution, but it is difficult to accurately calculate the pressure gradient to solve equation 2.4.

Eulerian methods discretise the field into cells, with the velocity and scalar quantities sampled at these fixed cell points in the space. This method is easier when calculating the pressure gradient, but can suffer from accuracy and resolution issues if the distance travelled by a parcel of air is less than the size of a cell, or significantly greater than the size of a cell.

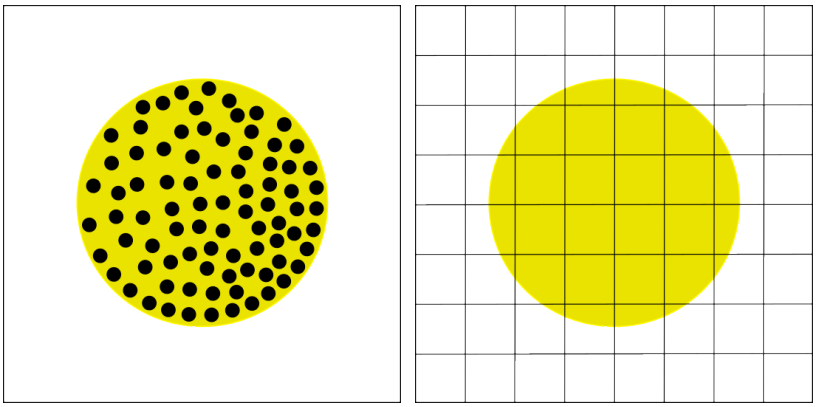


Figure 2.10: On the left, a Lagrangian approach. On the right, a Eulerian approach. In both cases, the fluid volume is indicated in yellow, with the outer square indicating the total volume.

2.4.4 Semi-Lagrangian advection

Solving the advection equation for a Eulerian method is unconditionally unstable[Bri15]. That is, there is no Δt that will not lead to a divergent result after some number of steps. Smaller values for Δt can increase the number of steps that are stable, but not indefinitely and this becomes impractical in short order. In comparison, the Lagrangian approach is unconditionally stable and solves the advection equation automatically as the particle is moved. However, it is possible to combine these two approaches, as was first done by Stam[Sta99].

Let us say we are examining the value of a quantity q at a grid point $\vec{x}_G$ at a time $t + \Delta t$. We can create a virtual particle P that is exactly at $\vec{x}_G$ at time $t + \Delta t$. To determine the value of q_P, we can trace P back to time t, when it would have been at position $\vec{x}_P$. We determine the position of $\vec{x}_P$ with the velocity field $\vec{U}$ at $\vec{x}_G$. By sampling the value of q at $\vec{x}_P$ at time t, we know the value of q_P, since equation 2.6 holds and is automatically solved as part of the Lagrangian treatment. Therefore, for each grid point $\vec{x}_i$ at time $t + \Delta t$, we can determine the value q_i by creating a virtual particle and tracing it back to time t. This method benefits from the spatial completeness and simpler solutions to equation 2.4 offered by the Eulerian approach, while still maintaining the unconditional stability of the Lagrangian approach.

2.4.5 Correcting for divergence

Applying the Navier-Stokes momentum equation (equation 2.3) on a discretised grid will violate the incompressibility condition (equation 2.4). This necessitates a correction step, which is the pressure term in equation 2.3 ($\frac{1}{\rho}\nabla p$). For a grid point $\vec{x}_i$, we can calculate the pressure gradient from the pressure values at neighbouring points using a central difference method:

$$(\frac{\partial p}{\partial x})_i \approx \frac{p_{i+1} - p_{i-1}}{2\Delta x} \tag{2.7}$$

Marker-and-Cell Grid

However, a naive implementation on a uniform grid can violate equation 2.6 if $p_{i-1} = p_{i+1}$ in equation 2.7, since the derivative could be 0 despite p not being

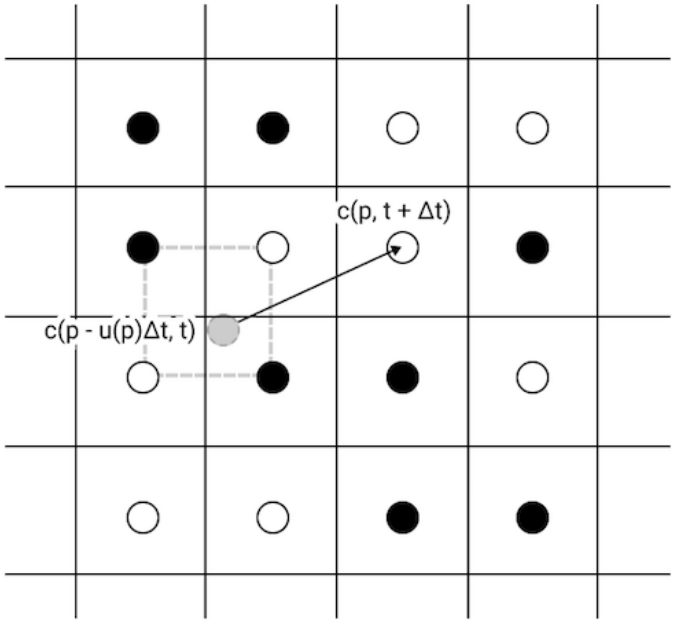


Figure 2.11: An example of semi-Lagrangian advection of a grid point at position p at time $t + \Delta t$.

constant across the grid. Alternatively, using p_i as one of the points in equation 2.7 will cause a bias towards one side, which is not desirable.

A solution to this is to use the Marker-and-Cell (MAC) discretisation, first proposed by Harlow and Welch[HW65]. This is a staggered grid where the velocity vector $\vec{U}$ is separated into its components and the components are placed on the edge of the cell, while scalar quantities including pressure are placed in the middle of the cell. See figure 2.12 for an example.

This grid discretisation is useful since it allows a pressure gradient calculation without discarding any values (as equation 2.7 does for p_i). The central differences method for a point $\vec{x}_i$ will then be:

$$(\frac{\partial p}{\partial x})_i \approx \frac{p_{i+1/2} - p_{i-1/2}}{\Delta x}, \tag{2.8}$$

which ensures that the derivative will only be 0 if $p_{i+1/2} - p_{i-1/2}$ is 0, i.e., $p_{i-1/2} = p_{i+1/2}$.

Pressure projection

Once the pressure gradient (∇p) is calculated, we project the effect of pressure onto the velocity vector field, with the constraint that equation 2.4 must hold afterwards. This gives us equation 2.9 for velocity at step $n + 1$.

$$\vec{U}^{n+1} = \vec{U} - \frac{\Delta t}{\rho} \nabla p \tag{2.9}$$

When applied to a two-dimensional MAC-grid (such as shown in figure 2.12), the velocity decomposes into u and v:

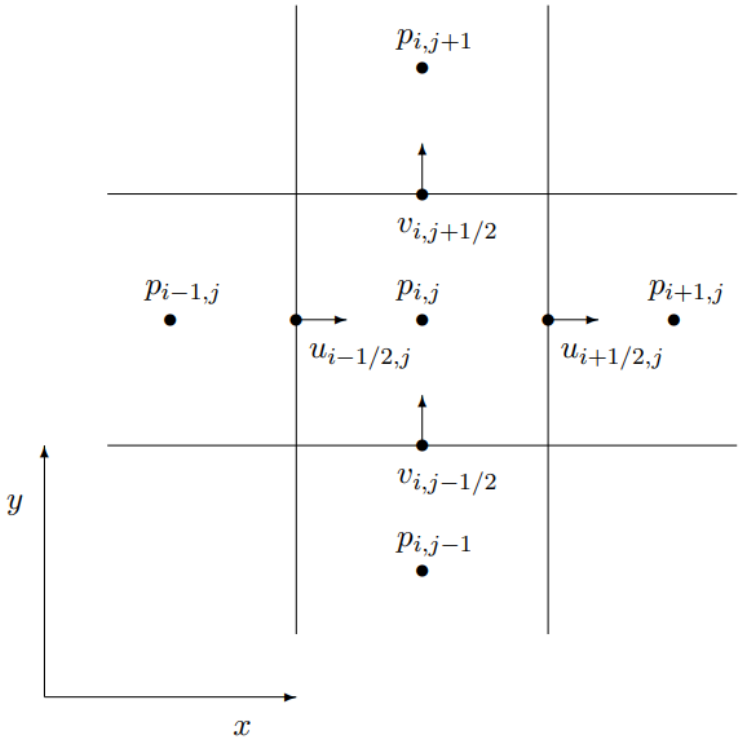


Figure 2.12: A standard Marker-and-Cell (MAC) grid representation, with velocity components (u and v) on the edges of the cells and pressures in the centre of the cell.

$$u^{n+1} = u - \frac{\Delta t}{\rho} \frac{p_{i,\,j+1/2} - p_{i,\,j-1/2}}{\Delta x} \tag{2.10}$$

$$v^{n+1} = v - \frac{\Delta t}{\rho} \frac{p_{i+1/2,\,j} - p_{i-1/2,\,j}}{\Delta y} \tag{2.11}$$

Calculating divergence

The pressure gradient is used to enforce equation 2.4. In two dimensions, the divergence of $\vec{U}$ is expressed as:

$$\nabla \cdot \vec{U} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \tag{2.12}$$

When applied to a MAC grid where the cells are square (i.e., $x \equiv y$), the divergence of the velocity in a cell i, j can be approximated using a central differences method as the following:

$$\begin{aligned} x &\equiv y \\ (\nabla \cdot \vec{U})_{i,\,j} &= \frac{u_{i,\,j+1/2} - u_{i,\,j-1/2}}{\Delta x} + \frac{v_{i+1/2,\,j} - v_{i-1/2,\,j}}{\Delta y} \\ (\nabla \cdot \vec{U})_{i,\,j} &= \frac{u_{i,\,j+1/2} - u_{i,\,j-1/2} + v_{i+1/2,\,j} - v_{i-1/2,\,j}}{\Delta x} \end{aligned} \tag{2.13}$$

Calculating pressure

At this point, no pressure value has been calculated yet. But by substituting equations 2.10 and 2.11 into equation 2.13, we obtain a system of linear equations (equation 2.14) where the pressures are unknown. See appendix A for the equivalent derivation when $\Delta x \neq \Delta y$, the square case is demonstrated here for legibility and comprehensibility.

$$\begin{aligned} \frac{u_{i,\,j+1/2} - u_{i,\,j-1/2} + v_{i+1/2,\,j} - v_{i-1/2,\,j}}{\Delta x} &= 0 \\ \frac{\left(u_{i,\,j+1/2} - \frac{\Delta t}{\rho}\frac{p_{i,\,j+1} - p_{i,\,j}}{\Delta x}\right) - \left(u_{i,\,j-1/2} - \frac{\Delta t}{\rho}\frac{p_{i,\,j} - p_{i,\,j-1}}{\Delta x}\right)}{\Delta x} + & \\ \frac{\left(v_{i+1/2,\,j} - \frac{\Delta t}{\rho}\frac{p_{i+1,\,j} - p_{i,\,j}}{\Delta x}\right) - \left(v_{i-1/2,\,j} - \frac{\Delta t}{\rho}\frac{p_{i,\,j} - p_{i-1,\,j}}{\Delta x}\right)}{\Delta x} &= 0 \end{aligned} \tag{2.14}$$

We can rearrange equation 2.14 to have the divergence of the cell (with inverse sign) on the right hand side:

$$\begin{aligned} \frac{\Delta t}{\rho}\frac{4p_{i,\,j} - p_{i+1,\,j} - p_{i-1,\,j} - p_{i,\,j+1} - p_{i,\,j-1}}{\Delta x^2} &= \\ -\left(\frac{u_{i,\,j+1/2} - u_{i,\,j-1/2} + v_{i+1/2,\,j} - v_{i-1/2,\,j}}{\Delta x}\right) & \end{aligned} \tag{2.15}$$

This forms a sparse linear system of the form $Ap = D$, where A is an $N \times N$ square matrix, where N is the number of cells in the system, and contains the coefficients of the various pressures in equation 2.15. Here, p is a column vector with N entries, as is D, where $D_i = -(\nabla \cdot \vec{U})_i$.

We then solve for p using any matrix solver of our choice. However, equation 2.15 is a symmetric positive definite matrix and this makes it very well suited to the conjugate gradient solver, as first described by Hestenes and Stiefel[HS+52]. Since we constrain our Δt such that air parcels do not travel more than a handful of cell widths at a time, the pressures in our system tend not to vary wildly between steps. This allows us to use the pressures calculated in the previous time step as the first guess for the next, which significantly speeds up convergence.

The pressures calculated can then be used with equations 2.10 and 2.11 to ensure $\vec{U}$ is nondivergent.

2.4.6 Boundary conditions

Domain boundaries

We model the domain boundaries using a wraparound scheme, creating a toroidal domain for each layer. For a domain of $N_x \times N_y$, the first column ($j = 1$) has as its neighbours the second column ($j = 2$) and the last column ($j = N_x$). Similarly, the first row ($i = 1$) has the second row ($i = 2$) and the last row ($i = N_y$) as its neighbours.

Solid boundaries

The equations presented in section 2.4.5 assumes that all cells in the system are fluid cells. However, a system with solid obstacles present require an additional constraint to the pressure calculation. Specifically, the velocity tangential to a solid interface must be equal to the velocity of the solid interface (equation 2.16, where $\vec{u}_S$ is the velocity of the solid interface).

$$\vec{u}^{n+1} \cdot \hat{n} = \vec{u}_S \cdot \hat{n} \tag{2.16}$$

In our case, the obstacles are in the form of terrain (e.g., mountains, cliffs, or hills impinging on the air layer). These are stationary, and therefore $\vec{u}_S$ will always be 0.

Since we use the MAC grid, we will always have a velocity stored on the exact grid point where a solid interface might be (since cells can either be fluid or solid). By replacing the velocity of some boundary in equation 2.13 with $\vec{u}_S$, a few generalisations become apparent. The coefficient in front of $p_{i,\,j}$ is the number of non-solid cells that border cell i, j, and any neighbour that is solid has its pressure term removed from that row of the matrix. Also, the velocity of the point in question on the right hand side (in the divergence calculation) is replaced by $\vec{u}_S$. See appendix B for an example derivation.

Chapter 3

Related work

Ever since technology progressed sufficiently to display digitally generated images, the ability to generate and display clouds and other weather phenomena has been considered a key part of digital landscapes on multiple scales. Like many advances in technology, the first steps in this direction were fuelled by military or pseudo-military applications. The first published work on visual cloud modelling is a report for the National Atmospheric and Space Administration (NASA) by Barnes et al.[Bar+68] in 1968, which used data from meteorological records to generate probabilistic cloud cover distributions on a global scale, aimed to assist in planning and training for the Apollo space missions.

Research in meteorology had been using numerical simulations for a few years by then, but the first report using a visual representation was by Mancuso et al.[MSB71] in 1971, incorporating a simple command line representation of the clouds from below and from a side-on perspective. A lot of research in this area followed, though the researchers' aims were generally to create a model that was as physically accurate as possible, giving less concern to the appearance of the clouds.

As late as 1983, the state of the art graphics used for NASA's Shuttle Mission Simulator did not have clouds for the in-atmosphere scenes, and only one configuration for clouds as seen from orbit[Hen83]. But with the proliferation of digital computer games during the 1980s, an increased focus was put on the visual appearance of clouds and skyscapes, especially if it could be done at a lower computational cost. Gardner[Gar85] built on the work of several contemporaries, such as Kajiya[KV84], to create a model for cloud generation focused on appearing as realistic as possible without having an exorbitant computational cost. The rate of progress has only increased since, with digital games, the entertainment industry, and meteorologists all requiring clouds of varying realism and physical accuracy, buoyed by the exponential increase in computing power available. However, to date, no one has addressed the extremes of temperature (up to 1300 K) and altitude (>20 km) found within volcanic plumes in a computer graphics application.

In contrast with the glut of work on clouds, volcanic plumes have languished, with little attention given to this topic within the computer graphics community. The methods presented have limited physical accuracy, especially with regards to the effects that interaction with the atmosphere present.

Scale	Procedural	Simulation	Data-driven	Hybrid
Micro-	[Ney97; Sch+03]	[KV84; MDN02a] [Har+03; Bi+16]	[Yua+14]	[Xu+09a]
Meso-	[Gar85; MB15] [Web+18]	[Dob+00; Miy+01] [FDY15]	[Sat+22] [Gos+23]	[Xu+09a; Xu+09b] [DG17; GN17; Gos19] [Häd+20; Vim+20]
Synoptic	[WM81]	[DYN06]	[Dob+99]	[Xu+09a]

Table 3.1: Cloud generation systems for skyscapes can be classified according to the scale at which the methods operate, and the broad category that the method falls under. While a lot of research has been devoted to micro- and mesoscale skyscapes, synoptic scales have received little attention. Data-driven and hybrid methods have become prevalent in recent years.

In this chapter, we examine the previous body of work in cloud animation and volcanic plume modelling, and highlight the shortcomings we address with this work.

3.1 Cloud modelling and animation

There are several methods of generating and animating clouds, each with their own advantages and disadvantages. We divide the methods into procedural, simulation, data-driven, and hybrid techniques. We also divide the scale on which proposed methods operate into the microscale (<1 km across), the mesoscale (<100 km across), and synoptic (global) scales. See Table 3.1 for a classification of proposed methods into these categories.

Once the clouds are generated and animated, care must be taken when rendering to achieve a realistic appearance. However, the particulars of cloud rendering is beyond the scope of this dissertation, and we refer the reader to Kallweit et al.[Kal+17] and Wilkie et al.[Wil+21] for a more comprehensive look at the subject.

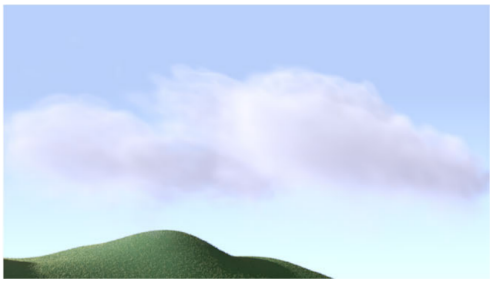
Because cloud modelling and animation is a mature research topic, a full account is impossible and this section will therefore cover the essentials only. See Goswami[Gos20] for a recent survey of the field.

3.1.1 Procedural

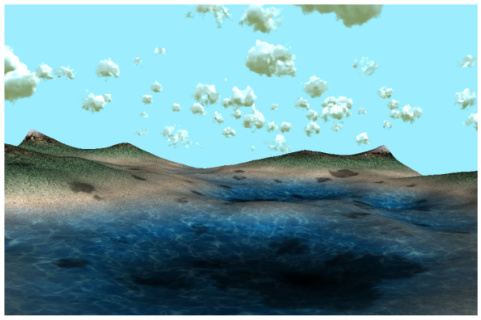
Procedural techniques generally focus on speed and quality rather than the physical accuracy of the clouds. These techniques generally aim to create a density field or function, which is then used to generate and render clouds. Given the origins of cloud modelling coming from climate simulation, most of the initial work in the field was focused on accuracy rather than computational cost. Procedural methods are generally fast, but they generally struggle to model cloudscapes over time and do not scale well.

Procedural noise functions are generally used to create the textures after the clouds are generated in some fashion. Gardner[Gar85] modelled cumuliform clouds

(1) [Web+18]



(2) [FDY15]



(3) [GN17]

(4) [Gos+23]

(5) [Vim+20]

(6) [Häd+20]

Figure 3.1: Examples from several works on cloud modelling and animation. (1) shows a procedural method from Webanck et al.[Web+18], while (2) and (3) show Lagrangian simulation methods from Ferreira Barbosa et. al[FDY15] and Goswami and Neyret[GN17], respectively. (4) shows a data-driven method from Goswami et al.[Gos+23]. (5) and (6) are hybrid methods from Vimont et al.[Vim+20] and Hädrich et al.[Häd+20], both employing simulation methods with procedural amplification.

using ellipsoids and blending multiples to form a cloud, while Neyret[Ney97] used bubbles to similar effect. Schpok et al.[Sch+03] used a combination of ellipsoids and particles textured with perlin noise to produce volumetric clouds that had realistic animation.

Mukhina and Bezgodov[MB15] used perlin noise to generate 2D stratocumulus clouds, using multiple layers to give the appearance of realistic, volumetric clouds. With rendertimes under 10 milliseconds per frame even on low-end hardware, it is very fast, albeit limited to only one type of cloud. Webanck et al.[Web+18] combined two procedural functions, one for shape and one for cloud presence, to compute density functions for multiple types of clouds, using primitives to model the cloud. They also allowed for users to define shapes for the clouds using a control field. However, their system encounters challenges with terrain interaction given the procedural nature of their cloud map. Figure 3.1(1) shows an example image from their system.

Procedural methods have the benefit of being very fast and very scaleable. However, because procedural functions have some degree of randomness, temporal and spatial consistency can be difficult to achieve over small timesteps. In addition, procedural methods are usually tuned to a specific cloud type, requiring different functions or parameters for different cloud types. The clouds generated by procedural methods have no guarantee of being physically accurate, which prevents coupling a procedural system on its own with other circulation models for different phenomena where bi-directional feedback is required. These methods are of significant use to enhance the results of other cloud generation methods, which we cover in section 3.1.4.

3.1.2 Simulation

Most simulations for cloud modelling rely on solving systems of partial differential equations (PDEs) based on the theories of fluid dynamics and Navier-Stokes equations. Simulations have the advantage of generally appearing realistic and being trivial to model dynamically over time and scalable to large volumes, at the cost of very long computation times, as well as offering little or no control. Simulation methods can usually be divided into Lagrangian (particle-based) and Eulerian (grid-based) methods.

Eulerian Eulerian methods have been used in a number of approaches. Miyazaki et al.[MDN02a] and Harris et al.[Har+03] both used a grid of voxels and a simplified model of fluid dynamics to model wind, humidity, and temperature. Dobashi et al.[DYN06] used a voxel-based grid to model a global cloudscape with user-provided pressure maps according to fluid dynamics. Xu et al.[Xu+09a] used a combination of grid-based fluid dynamics and metaballs to have realistic images at both global and small scales.

Lagrangian Kajiya and von Herzen[KV84] defined a system of PDEs to create a convective atmosphere where the humidity determines the formation of clouds,

using 'air parcels' advected over time. Ferreira Barbosa et al.[FDY15] created a Lagrangian model for cloud simulation, with particles splitting and merging depending on the interest level of an area for increased efficiency, building on the work of Adams et al.[Ada+07]. Goswami and Neyret[GN17] used a more traditional particle model to create clouds, with procedural generation of textures for a realistic look, but is limited in resolution and accuracy by the number of packets chosen. Figure 3.1(3) shows an example render from this system.

Cellular automata were used by Dobashi et al.[Dob+00], using discrete values, to efficiently simulate clouds and generate a density distribution. Miyazaki et al.[Miy+01] used real numbers in cellular automata to generate multiple cloud shapes that developed accurately over time. More recently, Bi et al.[Bi+16] used a combination of cellular automata and particle simulations to simulate clouds.

Simulation methods have the advantage of being physically accurate, temporally and spatially consistent, and capable of reproducing nearly all cloud types at sufficient resolution. This comes at a cost of performance, with simulation usually being in the interactive range or slower, only rarely being faster than real time. It also carries the cost of complexity, usually requiring a full computational fluid dynamics treatment. Comparing different simulation methods, Lagrangian methods generally result in better peak resolution than Eulerian methods. However, this comes at a cost of some areas of the domain potentially having a very poor resolution.

3.1.3 Data-driven

Data-driven techniques rely on real data to generate or modify a model. Dobashi et al.[Dob+99] used satellite images to create a metaball-based model. Dobashi et al.[DSY10] modelled cumulus, altocumulus, and cirrus clouds by extracting features from photos of real clouds. Yuan et al.[Yua+14] refined this work for cumulus clouds, generating a more detailed three-dimensional representation of the cloud. More recently, Satilmis et al.[Sat+22] used a convolutional neural network (CNN) trained on 630 images of real skies to create skymaps from user provided cloud maps and sky settings. Goswami et al.[Gos+23] utilised the growing trend of convolutional generative adversarial networks (CGANs) to animate cloud maps generated from a model trained on ground-based videos of real cloudscapes. However, these animations do not have the fine detail present in real clouds. Figure 3.1(4) depicts an example from Goswami et al.'s system.

Data-driven models suffer from a number of drawbacks. Firstly, acquiring a sufficiently large dataset that is labelled is challenging at best. Satilmis et al.[Sat+22] used a very limited number of images for their method, much less than what is standard in machine learning literature. This can lead to temporal artefacts, or limiting the length of animation to avoid said artefacts. To boot, acquiring volumetric data for datasets is both technically challenging and expensive. With only one planet's example of real data, it is challenging to generalise to different planetary bodies. Data-driven models do have the benefit of being fairly fast (after the model is trained), and requiring fairly little in terms of initial conditions and parameter tuning (once the model is trained).

3.1.4 Hybrid models

It has become increasingly common for modelling systems to incorporate a hybrid method combining two or more of the techniques mentioned above. Xu et al.[Xu+09b] combined a simulation using cellular automata to determine the density field and procedural generation with Brownian motion functions for the fine shape of the clouds. Goswami and Neyret[GN17] used a Lagrangian fluid dynamics simulation at a large scale and procedural hypertextures on smaller scales for computationally inexpensive yet realistic-looking clouds. This work was extended by Goswami[Gos19] for more realistic cloud distributions and improved performance, albeit limited to only a single layer.

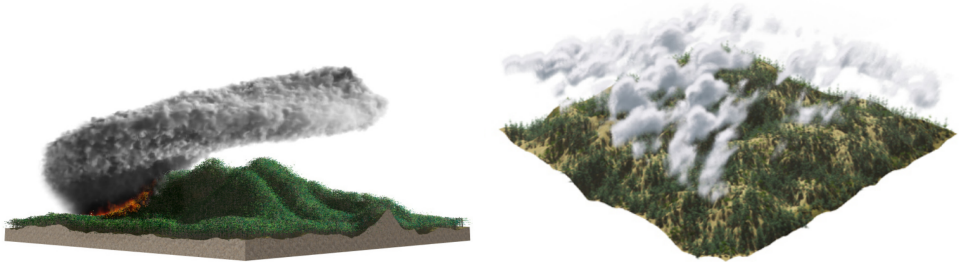
Duarte and Gomes[DG17] used a simplified Lagrangian simulation combined with thermodynamic diagrams from real data to simulate clouds in real-time. Vimont et al.[Vim+20] used a layered Eulerian meso-scale simulation to simulate weather at a coarse scale, combined with procedural noise amplification at finer scales to generate a wide array of cloud types at interactive rates. However, this system uses a procedural method to couple the layers, which can lead to unrealistic behaviour. Figure 3.1(5) shows an example render from Vimont et al.'s method. Hädrich et al.[Häd+20] combined a detailed physical simulation with procedural amplification of heat and wind maps to create intricate clouds of many different types, including several variations of cumulonimbus seen in nature. However, the underlying heat maps require parameters set by the user, and this requires in depth knowledge to accurately create the desired cloud types. Figure 3.1(6) shows an example render of a cumulonimbus cloud in Hädrich et al.'s system.

Hybrid models have a benefit over procedural by generally being more accurate over time, as well as modelling a wider range of cloud types. They tend to require less time than pure simulation methods for the same fidelity. However, this comes at the cost of significant complexity, and an increased overhead for tuning the parameters and setting the correct initial conditions for the desired scenario.

3.1.5 Combined models

Of particular interest to us are methods where two or more submodels for different phenomena interact for a combined simulation, in contrast to hybrid models which utilise two or more methods for generating clouds (a single output). Global circulation models (GCMs) are common in climate science, and combine atmospheric, ecological, and oceanographic models to cover as many aspects of the global climate as possible. However, these models are extremely complex, and require immense computation power. For example, Wedi et al.[Wed+20] simulated four months of global circulation using 5 days of compute time on OLCF-4/Summit[Hin18]. This is not feasible for graphical applications (with some exceptions, discussed in section 3.1.6), and we therefore focus on smaller scale simulations for graphics applications.

Hädrich et al.'s Fire in Paradise[Häd+21] combines a module-based ecosystem model with a hybrid atmospheric system drawn from their previous work on stormscapes[Häd+20] and extended to include rain vaporisation. This method models the spread of fire, smoke, and clouds to depict forest fire dynamics. Complex feedback



(1) [Häd+21] (2) [Pał+22]

Figure 3.2: Example renders from two combined models. (1) depicts a flammaginetus cloud from Hädrich et al.'s Fire in Paradise[Häd+21], where a forest fire leads to convection and subsequent condensation. (2) is an example render from Pałubicki et al.'s Ecoclimates[Pał+22], which combine terrain and vegetation submodels with an atmospheric model.

effects arise from the two-way interaction between the fire sub-model and the atmosphere. Flammaginetus clouds form in strong convective zones above the hot fires, leading to uplift. This uplift can cause water vapour to condense, leading to rainfall which can extinguish fires again. Hädrich et al. successfully model forest fires under different conditions, including conditions that lead to flammaginetus clouds forming in convective columns. The authors do not provide detailed computation times, but claim interactive rates in all scenarios. They are limited in resolution to the microscale as per the definitions at the start of this section. However, the authors refer to their system as operating on the mesoscale, based on the processes modelled not being at particle size.

Ecoclimates from Pałubicki et al.[Pał+22] provide a system capturing the bi-directional feedback cycles between the atmosphere and the ecosystem that result in microclimates and specific vegetation/biome patterns. The authors use temperature and moisture maps to exchange values from the atmospheric and the soil and vegetation sub-models. Reducing precipitation leads to reduced vegetation cover, which leads to reduced moisture uptake through transpiration. The system runs at interactive rates, at between 0.5 and 8 seconds per step (where each step is one simulated year). The resolution is fairly small at 20 metres on a side for the voxels of the atmospheric grid. They do not take diurnal or seasonal changes into account, and it suffers from a poor time resolution (each time step is a simulated year) for the purpose of cloud animation.

Like our system, both Hädrich et al. and Pałubicki et al.'s simulations are tailored to very specific phenomena. Similar to hybrid models, combined models have potential for increased physical accuracy, as well as modelling phenomena that are not available with single-system models. However, this comes at the cost of significant complexity with combining multiple models, as well as increasing the overhead for tuning parameters and initial conditions. These combined models generally also increase the computation time required.

3.1.6 Synoptic scales

Most research on cloud modeling in computer science has focused on micro and meso-scales, sufficient for many applications, but synoptic (global) scale modeling has distinct advantages, such as for planetary visualization or providing boundary conditions for smaller-scale models. While this work focuses on meso-scale skyscapes, it has the potential to be expanded to synoptic scales and a mention of salient work in that field is warranted. Early work by Wetherald and Manabe[WM81] modeled global clouds using atmospheric parameters, though it faced limitations with seasonality and calibration. Dobashi et al.[DYN06] created an Eulerian simulation for global cloud generation and later improved it with a rendering system[DYN10] for multiple scales. Xu et al.[Xu+09a] further combined large-scale simulations with smaller-scale cloud amplification, though this method lacked fine detail and terrain interaction. Microsoft's Flight Simulator (2020)[SM20] uses real-time weather data and global circulation simulations for high resolution skyscapes across the globe, but its heavy reliance on data and cloud computing limits its application to Earth's weather modeling.

3.2 Volcanic plume

In stark contrast to the body of work dedicated to cloud generation in graphics, there has been little focus on the generation of volcanic plumes. All previous work for plumes in the graphics domain focused solely on the volcanic plume, with no regard to atmospheric phenomena arising from the plume interactions. In addition, all previous work utilised a uniform wind field that could, at best, be varied as a function of altitude, along with a static atmosphere for entrainment interactions. Mizuno et al.[MDN02b] provided the first simulation of the volcanic plume for graphical applications, using a full 3D Eulerian grid combined with semi-Lagrangian advection. Using a grid of $100 \times 100 \times 100$ voxels, they perform a simulation of the volcanic plume based on the Navier-Stokes equations for fluid dynamics.

By applying procedural buoyancy, "vorticity confinement", and wind forces before advection, their system produces plumes that appear relatively realistic. This procedural basis allows their system to retain a fair degree of control over the plume by varying the parameters of the eruption, such as a narrower, taller cone if the eruption is more powerful. However, their resolution is coarse, with little or no fine detail being produced. Despite using an efficient Coupled Map Lattice solver for the Navier-Stokes pressure projection, performance is slow. However, a more modern implementation on a GPU would result in a significant performance increase. The system is limited to the plume, ignoring related atmospheric phenomena, and the uniform wind field is not physically accurate. The procedural nature of the buoyancy forces leads to artefacts and behaviour that isn't physically accurate, nor does it account for turbulent effects arising from the interaction of the plume with the atmosphere. Figure 3.3(1) depicts an example render from Mizuno et al.'s system, utilising a rendering system created by Dobashi et al.[DYN02].

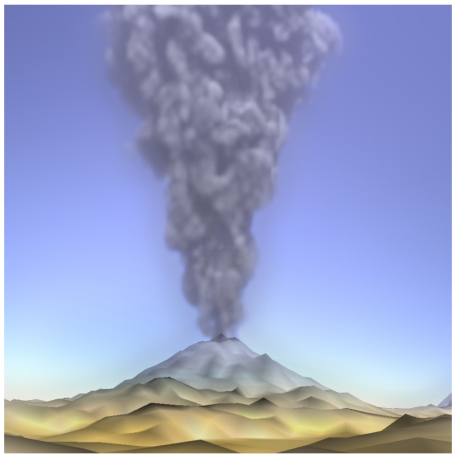
To address some of the issues arising from the procedural nature of Mizuno et al.'s[MDN02b] plume dynamics, Mizuno et al.[Miz+03] created a model that simu-

lates the air and the hot ejecta from the eruption as two separate fluids. The hot ejecta (referred to as "magma" by Mizuno et al.) is the mix of pyroclasts and hot gas that is erupted. This magma has an initial velocity and entrains atmospheric air as it rises. A buoyancy force is calculated based on the density in a voxel, which in turn is determined by the temperatures and masses of the atmosphere and the magma components in the voxel. For performance, Mizuno et al. make the simplifying assumption that the temperatures remain constant, since that allows the density calculation to be reduced to a preprocessed lookup table. Using a grid of $150 \times 150 \times 150$ voxels, Mizuno et al. simulate volcanic plumes in greater variety than the earlier work ([MDN02b]) merely by increasing or decreasing the temperature difference between the two fluids. This approximates the force of the eruption, leading to noticeably different behaviour in the plume. This work can generate physically accurate and realistic shapes for volcanic plumes across a range of eruptions. However, the performance is slow despite the simplification, and this limits resolution. Like with the earlier work, this system cannot generate fine detail due to the coarse resolution of the voxels (each voxel being 100 m on a side). The system is limited to the volcanic plume and does not take atmospheric phenomena into account. Also, similar to the earlier work, Mizuno et al. use a uniform, procedural wind field, which is not physically accurate. Figure 3.3(2) shows a plume generated by this system.

In a departure from earlier work, Lastic et al.[Las+22] proposed a model utilising cylinders following Lagrangian dynamics stacked along the centre line of the volcanic plume. These cylinders are enhanced by a hierarchy of rotating spheres and subspheres, which are used in a procedural upsampling step to introduce fine resolution detail. Our model is partly inspired by this work and we direct readers to chapter 5 for more detail on the implementation of this model. Lastic et al.'s model can generate a variety of volcanic eruptions, and is the first to explicitly consider pyroclastic flows and subsequent minor plumes for graphics. Combined with a fair degree of control available to the user and impressive simulation performance, it can be considered the state of the art. However, like its predecessors, it relies on a uniform windfield, and does not consider any interaction with a dynamic atmosphere.

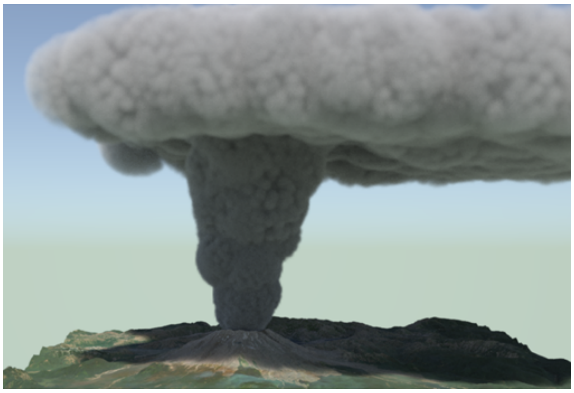
While not directly related to volcanic plumes, work on simulating explosion plumes[YOH00; KK14] and smoke requires similar examinations of entrainment as occurs in volcanic plumes. A full review of the field is beyond the scope of this work and we direct the reader to a review by Thuerey et al.[TKP13].The majority of work on smoke in computer graphics used Eulerian grid-based methods[KV84; FM97; Sta99; FSJ01; WB07]. These have much the same characteristics as the work on volcanic plumes by Mizuno et al.[MDN02b; Miz+03], with computationally intense simulations at fairly coarse resolutions, aiming to model the physical processes resulting in smoke. Shi and Yu[SY05], following up on work by Treuille et al.[Tre+03] and Fattal and Lischinski[FL04], provide an interesting contrast with a system to constrain smoke (generated by a fluid simulation) to transform into specific, user-selected shapes.

Several extensions aimed at reducing the computation required for realistic smoke have been devised, such as moving the simulation grid to follow the smoke[Sha+04; Zar20], or using dynamic grids[Pat+05]. As with cloud simulation, using a coarse



(1) [MDN02b]

(2) [Miz+03]



(3) [Las+22]

Figure 3.3: Example renders from three models of the volcanic plume. (1) shows a render from Mizuno et al.[MDN02b] using a 3D simulation with procedural buoyancy forces. (2) shows a render from Mizuno et al.'s[Miz+03] modelling the air and hot ejecta as two different fluids. (3) gives an example from Lastic et al's[Las+22] system using Lagrangian slices and a hierarchy of spheres and subspheres for procedural upsampling.

simulation with procedural upsampling for detail has been used for smoke[SB08]. Recently, using a machine learning approach for both solving and upsampling fluid simulations for smoke animation has become an extremely active field[CT17; Xie+18; EUT19; Kim+19; Wer+19; Dry+22; Den+23]. This is by no means an exhaustive review, and reviews of this narrow field by Chen et al.[Che+21] and Giraldi et al.[Gir+23] can be consulted.

3.2.1 Geophysical simulations

Compared to the relative dearth within the graphics research, volcanic eruptions and the associated plumes has been an area of intense research in the geophysical sciences. These works typically model the plume as a multiphasic flow, with the domain discretised as a regular 3D grid[DNM93; Obe+98; DR06; Ong+07; CEB16; Ner+21]. This is akin to the system described by Mizuno et al.[Miz+03], though with a much larger domain and significantly more accurate physics models. This extends to the inclusion of a much greater range of effects, including but not limited

to volcanic jets, collapsing plumes, ballistic ejecta, and convective plumes. For a comprehensive review of the field, we direct the reader to a review by Kavanagh et al.[KEM18].

In addition to plume models, related atmospheric phenomena has been studied in detail. Volcanic eruptions are responsible for the transport of significant quantities of water[GBW97; Mil+22; Kha+22], which can have effects on cloud formation in the local area, as well as long term effects on a global scale. The entrainment of ambient air into the plume, the crucial component that dominates the behaviour of all plumes, is examined in detail. Morton et al.[MTT56] provided the first model of entrainment, which has informed all plume and most smoke models since. Work is continuing in this field, with recent models focusing on microscale interactions within the plume[SK10; Bon+15; CON16].

The aggregation of ash and other particles within the plume, which we use to inform our ash rain implementation, remains an active area of research[GL94; BBD12; Hof+23]. This aggregation contributes significantly to the formation of volcanic lightning, which remains under investigation both in modelling and in laboratory conditions[Jam+08; Cim+14; Van+16; McK+21; Smi+21; CG22]. Many of these models aim to characterise the relationship between the amount of lightning and the amount of ash ejected by the volcano.

The models produced in this field, while as physically accurate as possible, are not suitable for use in computer graphics. The computation required for the physical accuracy in these models is exorbitant, often requiring days or weeks to execute, even on hundreds of CPU cores[Suz+16]. With a large domain (up to $350km \times 350km \times 50km$), the grid sizes are still large, ranging from 300 metres to 4 000 metres on a voxel side. While this is sufficient for chemical composition and thermodynamic analysis[Tex+05; Suz+16], it cannot replicate the fine-scale turbulence that is the defining visual characteristic of volcanic plumes.

However, these studies serve as a rich source for validating our contribution, as described in section 7.1. In particular, evaluating the accuracy of the emergent properties of our system, such as ash deposition, plume shape, and vertical velocity, benefits greatly from the work in the geophysical sciences.

3.3 Summary

Significant effort has been dedicated to the modelling and animation of clouds within computer graphics, at multiple scales and with a wide variety of methods. Within the last 10 to 15 years, hybrid methods have become particularly prevalent with results notable for the computational performance, and the visual accuracy and variety of clouds generated. With the advent of machine learning and a variety of generative models, the future of cloud animation will likely be with hybrid models incorporating some degree of generative upsampling. Importantly for this work, no previous cloud modelling method has dealt with the extremes of temperature and altitude that volcanic plumes generate.

In contrast, volcanic plumes on their own have received little attention, with only three previous methods in computer graphics. Notably, all three previous methods utilised a static atmospheric boundary, something that this work aims to address.

Chapter 4

Weather modelling

In this chapter we explain our model of the atmosphere and its physics, how transport within the atmosphere is modelled, and how the different layers in the atmosphere exchange quantities.

There were a number of different options available for modelling the atmosphere, as detailed in section 3.1. While the choice of plume model was simple given the minimal options presented, the weather and cloud model presented more variety. We required a system that was performant, produced realistic clouds, and that could be coupled with a volcanic plume solution. This last requirement immediately eliminated the procedural methods (section 3.1.1), and severely constrains the potential for data-driven methods (section 3.1.3).

Examining the options for simulation (section 3.1.2) or hybrid models (section 3.1.4), Vimont et al.'s system[Vim+20] stood out – it is fast, it produces realistic clouds, and utilises a coarse simulation with concrete values for temperature and moisture, making it suitable for a coupling with Lastic et al.'s volcanic plume simulation[Las+22].

Examining the source code provided by Vimont et al., we found that much of their interactions are dependent on a hard-coded grid structure. However, given that most operations within their system occur on a per cell basis, this dependency is unnecessary and hinders adaptations of their code. Therefore, we reimplemented their system to remove this dependency. As part of the reimplementation, a major issue with their fluid dynamics treatment was encountered. While very fast and efficient, it does not take solid boundaries into account. This results in significant inaccuracies.

Therefore, we implemented our own fluid dynamics treatment (described in section 2.4), which adds a significant computation cost, but does provide much more realistic fluid behaviour. Despite the added cost from our fluid dynamics treatment, combined with optimisations and parallelisation in other components our implementation still showed a two to three times speedup compared to that of Vimont et al.

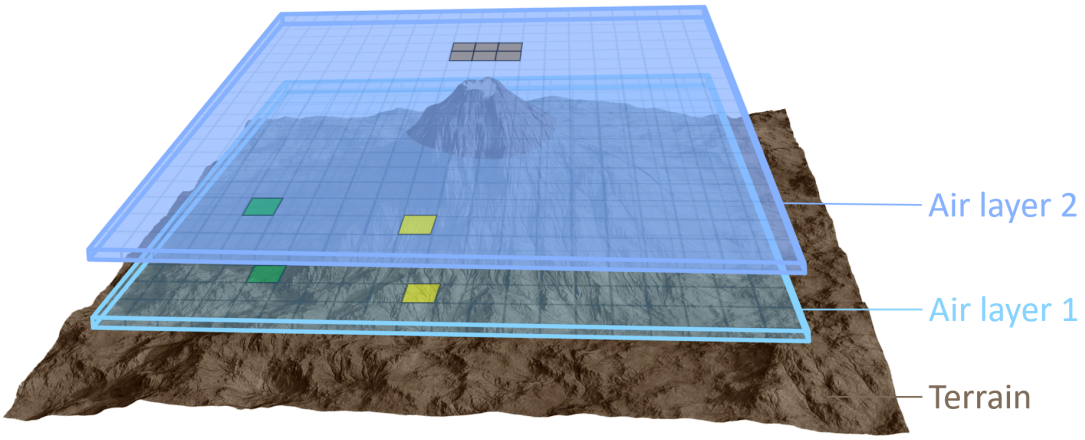


Figure 4.1: The layered structure of the atmospheric model, with two air layers and cells within these layers. The cells coloured green and yellow indicate cells that are vertically adjacent to each other. The cells in air layer 2 that are coloured brown indicate some of the cells immediately above the terrain, where the terrain impinges on air layer 1.

4.1 Model structure

We model the troposphere (that is, the lowest layer of the atmosphere) as several 2D fluid layers, recognising the tendency for the atmosphere to be separated into distinct horizontal layers. Each cell in a layer consists of a velocity vector and several scalar quantities as detailed in table 4.1. These quantities need to be transported accurately according to the local wind field, represented by the 2D velocity vector.

To complete our atmospheric model, we have vertical exchange layers between each horizontal layer to model the interaction between the different horizontal layers, and a single surface layer representing the terrain. Figure 4.1 shows the layered nature of our model. The cells coloured green and yellow indicate cells that are vertically adjacent to each other. The cells in air layer 2 that are coloured brown indicate some of the cells immediately above the terrain, where the terrain impinges on air layer 1.

A full overview of all components and quantities transferred by the components can be found in figure 4.2, while algorithm 1 shows the substeps within a single time step of the complete weather subsystem.

4.1.1 Horizontal air layers

Each horizontal air layer consists of a velocity field, separated into components, a temperature scalar field, a moisture scalar field, a condensed water (cloud) density scalar field, a particulate density field, a (modified) pressure correction scalar field, and a mask dictating where the terrain impinges on the air layer. The velocity, pressure correction , and terrain mask fields are required for advection and describing the movement of the atmosphere, with the component-wise split of the velocity described and motivated in section 2.4.5. The temperature, moisture, cloud density, and particulate density fields dictate the real-time appearance of the atmosphere.

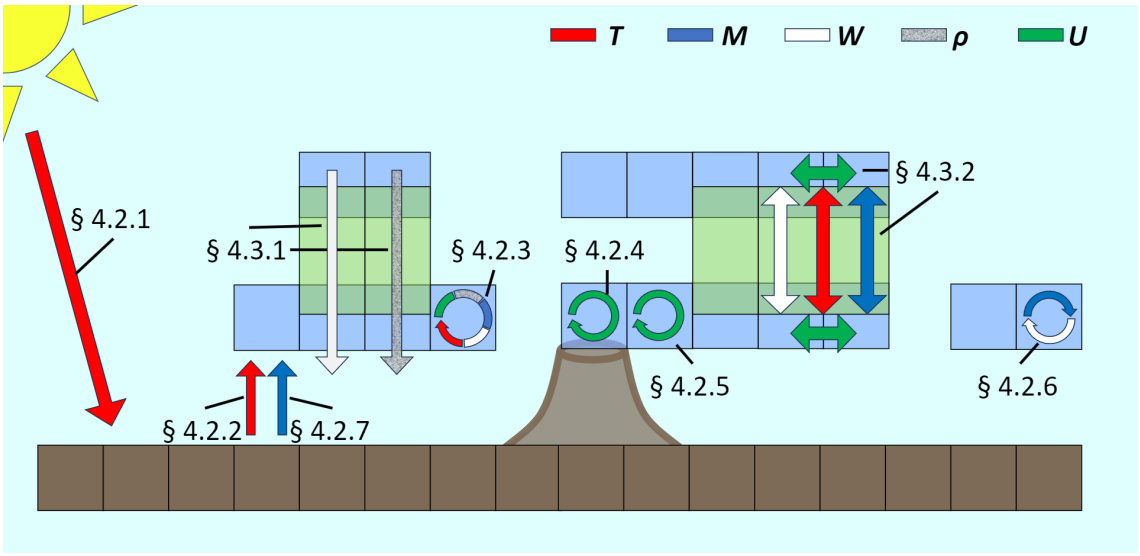


Figure 4.2: The weather subsystem consists of several different components. The different quantities are transferred between cells within a layer, or between vertically neighbouring cells in different layers, with the arrows indicating the direction of transfer. Temperature T is red, moisture M is dark blue, water W is white, ash density ρ is textured grey, and the 2D velocity U is dark green, as indicated by the legend at the top of the figure. The terrain model is brown, while the horizontal air layers are light blue and vertical exchange layers are light green. From left to right, in the order that the engines operate within a full time step, we have the terrain heating engine (§4.2.1), the atmospheric heating engine (§4.2.2), the moisture uptake engine (§4.2.7), and the precipitation engine (§4.3.1). The advection engine (§4.2.3), and obstacle friction engine (§4.2.4) are independent per cell in a layer, while the pressure correction engine (§4.2.5) calculates prerequisite values independently per cell before the matrix solve. The vertical coupling engine (§4.3.2) and the phase transition engine (§4.2.6) are the final components of the subsystem.

We simplify all particulate matter in the atmosphere to be either liquid water or particulates, neglecting both ice and the wide array of sizes that particulate matter can take with resulting differences in visual appearance. We use this simplification as a compromise between visual accuracy and computational performance. A full atmospheric simulation requires prohibitive amounts of computation while adding details on the micro-scale, which would be lost in our grid discretisation.

The pressure correction field is used only in the pressure correction step (see section 4.2.5) and will not be referred to again in this section. Both the pressure correction field and the obstacle mask are dimensionless quantities. See table 4.1 for the components of each air layer and the symbols used to refer to them in the rest of the chapter.

Name	Symbol	Unit
North-south velocity	u_x	metres per second $(m \cdot s^{-1})$
East-west velocity	u_y	metres per second $(m \cdot s^{-1})$
Temperature	T	kelvin (K)
Moisture	M	kilogram per cubic metre $(kg \cdot m^{-3})$
Condensed water	W	kilogram per cubic metre $(kg \cdot m^{-3})$
Particulate density	ρ	kilogram per cubic metre $(kg \cdot m^{-3})$
Obstacle mask	O	–
Pressure correction	P	–
Height above reference	h	metres (m)
Thickness	δ	metres (m)

Table 4.1: A list of components of each horizontal air layer, their symbols, and their units. Since both the obstacle mask and the pressure correction field are dimensionless quantities, no unit is applicable.

The fluid dynamics are applied only to these horizontal layers, where they are modelled as 2D planes. In the greater model, however, they are 3D volumes with a thickness δ. The height above (or below) a reference height (which is the average sea level, in our case) refers to the middle of the air layer's thickness. The layers do not have a uniform thickness, instead the thickness depends on where the neighbouring layers are with the distribution of layers being determined by the user.

The user provides a maximum number of layers, a maximum height, and a list of heights that are of interest, i.e., would benefit from having a horizontal air layer at that height. For the scenarios described in chapter 7, the maximum number of layers is 20, the maximum height is 20000 metres, and the list of heights provided is $\{400, 1200, 2000, 2700, 3000, 3500, 4100, 4700, 5500, 6500\}$, with those heights being in metres above sea level. The final position and thickness of air layers is determined at initialisation, placing layers as close to the heights prescribed by the user as possible. In order to have air layers that are equally thick above and below the layer height, some adjustment might be required if the user's list of heights are not equally spaced. Appendix C shows an example initialisation, based on the values used in the scenarios described in chapter 7.

Name	Symbol	Unit
Vertical velocity	uz	metres per second $(m \cdot s^{-1})$
Rainfall	η	kilogram per cubic metre $(kg \cdot m^{-3})$
Particulate fall	ζ	kilogram per cubic metre $(kg \cdot m^{-3})$
Lower height	H_{b}	metres (m)
Upper height	H_{t}	metres (m)

Table 4.2: A list of components of each vertical exchange layer, their symbols, and their units. The first three components are per cell, while the lower two components are the same across the layer.

4.1.2 Vertical exchange layers

For complete physical realism, a full 3-dimensional grid with advection and pressure solves is required. However, this comes at a prohibitive computation cost, with a single solve of a $512 \times 512 \times 20$ grid requiring upwards of 100 times the compute time of 20 solves of a 512×512 grid. In addition, for optimal accuracy, the resolution in the z direction would need to be significantly increased, further increasing the computation cost. To avoid this substantial cost, we instead transfer quantities between horizontal layers and model the interaction between them in a procedural manner.

The vertical exchange layers are used to transport quantities between horizontal layers. This is done through the precipitation engine (section 4.3.1) and vertical coupling engine (section 4.3.2). Each vertical layer stretches from the middle of the horizontal layer below it to the middle of the horizontal layer above it. Since the horizontal layers can have differing heights, the reference height of the exchange layer does not have to lie on the border between the horizontal layers. See figure 4.2 for a diagrammatic representation, with the horizontal air layers in blue and the vertical exchange layer in green.

Each vertical exchange layer has three scalar fields. Firstly, the vertical velocity scalar field, as calculated by the vertical coupling engine (section 4.3.2), and used by the precipitation and vertical coupling engine. Then there are the rainfall and particulate fall scalar fields, which are used to track the precipitation of water and particulates respectively (see section 4.3.1). Table 4.2 lists the components of each vertical exchange layer and their symbols used in the reset of this chapter.

4.1.3 Terrain model

The terrain is treated as a single layer with varying elevation, ranging from the lowest depth below the ocean to the highest peak on land. Each terrain cell has an elevation, a temperature, a moisture content, and a terrain type. The terrain type informs a number of values such as albedo, emissivity, and heat capacity, with the different types being dictated by the planet model. In our case, we utilise the values of the Earth/Sun system. See tables 4.4 and 4.5 for details. In our implementation, terrain cells only interact with the air layer immediately above it, which depends on the elevation of the cell. This can be extended by other models, such as an

Name	Symbol	Value for Earth/Sun system	Unit
Radius	R	6 371 000	metres (m)
Sidereal rotation period	Ω	86 164.1	seconds (s)
Average temperature	$\bar{T}$	288.15	kelvin (K)
Reference height	h_0	0	metres (m)
Molecular mass of dry air	M_{air}	0.02896	kilogram (kg)
Lapse rate of dry air	Γ	0.098	kelvin per metre ($K \cdot m^{-1}$)
Specific heat capacity of dry air	c_{A}	1006	joule per kilogram per kelvin ($J \cdot kg^{-1} \cdot K^{-1}$)
Specific heat capacity of moisture	c_{M}	≈ 4000	joule per kilogram per kelvin ($J \cdot kg^{-1} \cdot K^{-1}$)
Latent heat of vaporisation	ΔH_{vap}	2257	joule (J)

Table 4.3: A list of quantities and constants for the planet, their symbols, their units, and their values in the Earth/Sun system.

Name	Symbol	Unit
Terrain type	X_i	See table 4.5
Elevation	h	metres (m)
Moisture	M	kilogram per cubic metre ($kg \cdot m^{-3}$)
Temperature	T_i	kelvin (K)

Table 4.4: A list of quantities and constants carried by each cell of the terrain, their symbols, and their units if any.

ecosystem model.

The planet model contains all the planetary constants, such as the lapse rate (that is, the averaged rate of change in temperature as a function of altitude), the gravitational acceleration at the surface, or radius. These can be changed by the user on initialisation. Table 4.3 contains a list of the planetary constants used in the system, while table 4.5 lists the values associated with each terrain type. The user can define any number of terrain types at initialisation, and a given cell can change its terrain type over time.

Name	Symbol	Unit
Terrain index	X	–
Density	ρ	kilogram per cubic metre ($kg \cdot m^{-3}$)
Specific heat capacity	c_{ter}	joule per kilogram per kelvin ($J \cdot kg^{-1} \cdot K^{-1}$)
Albedo	α	–
Emissivity	ϵ	–
Is dry	d_{X}	–

Table 4.5: A list of quantities for each terrain type, their symbols, and their units if any.

4.2 Physics within the atmosphere

All energy on Earth derives either from the Sun, or in rare cases, from left over heat from the formation of the planet or from the decay of radioactive elements. While the latter two are negligibly small in comparison to the former, they are generally thought to be a contributing cause to volcanic eruptions, and are therefore of interest as well.

Energy modelling can be divided into two broad categories. First is the addition and removal of energy in the entire system, i.e., energy from the sun added to the planet and atmosphere, and energy removed through dispersion and dissipation. The second category is energy storage and release within the atmosphere, i.e., energy used or released during phase transition.

In our case, since our time domain is limited to a few hours at most, the effect of terrain heating is limited as well. However, for any longer term simulations, this effect is of crucial importance and we therefore include it in our system.

4.2.1 Terrain heating

Since the atmosphere is largely transparent to light in the visible spectrum (where the sun radiates the majority of its energy), we make the simplification that the atmosphere absorbs no heat from the sun directly. This leaves heating of the terrain as the only avenue for energy from the sun to be added to the system. The Earth's surface absorbs some of the energy from the sun and then radiates it out in the infrared. The atmosphere is opaque to infrared radiation and absorbs some of it.

The energy that is absorbed by a given area of the terrain depends on the location of the sun relative to the terrain, the thickness of the atmosphere relative to the planet, how bright the sun is, the specific heat capacity of the terrain, and the albedo of the terrain. The albedo is the ratio of incoming light that is reflected instead of absorbed. For example, fresh snow has a very high albedo (0.95), reflecting the vast majority of light that hits it. In comparison, water has a low albedo (0.2), absorbing the majority of incident light.

We model the change in temperature for a terrain cell i as follows:

$$\Delta T_i = \frac{\Delta Q_i \Delta t}{c_{\text{ter}} \rho}, \tag{4.1}$$

where ΔQ_i is the energy per square metre falling on i, and c_{ter} is the specific heat capacity of the terrain. Note that c_{ter} is calculated as the proportional sum of the actual terrain and any water it might contain. Dry terrain has a lower specific heat capacity than moist terrain.

ΔQ_i is calculated as:

$$\Delta Q_i = L_i \cdot \cos Z_i (1 - \alpha_i), \tag{4.2}$$

where L_i is the solar intensity at the surface, and Z_i is the zenith angle (i.e., angle from the zenith to the sun) for i[1]. The solar intensity is approximated using the empirical formula below[MM76], where AM is the air mass coefficient for i, λ is the solar intensity in a vacuum at a distance of 1 astronomical unit (AU), and D is the distance between the sun and the planet in AU. For the Earth-Sun system, $\lambda = 1350\ W \cdot m^{-2}$ and $D = 1$ by definition.

$$L_i = 1.1\frac{\lambda}{D} \cdot 0.7^{AM^{0.678}} \tag{4.3}$$

Using the radius of the Earth as r, the air mass coefficient is calculated with the following equation[Sch29]:

$$AM = \sqrt{(r\cos Z_i)^2 + 2r + 1} - r\cos Z_i. \tag{4.4}$$

We also model heat loss through radiation to outer space, again using equation 4.1 to calculate heat lost. In this case, ΔQ_i is calculated as follows:

$$\Delta Q_i = \epsilon_i \sigma \left({T_i}^4 - {T_\Omega}^4\right) \times R\left(1 + \frac{h_i}{h_A}\right), \tag{4.5}$$

where ϵ_i is the emissivity of the terrain in cell i, σ is the Stefan-Boltzmann constant, and T_i is the temperature of the terrain in cell i. T_Ω is the background temperature of the universe ($2.7K$), and R is a user parameter determining how much of the energy escapes, roughly analogous to the chemical composition of the atmosphere. h_A is the height of the atmosphere, and this ratio is applied to R to model how the atmospheric density decreases as the elevation increases.

4.2.2 Atmospheric heating

After the terrain is heated by absorbing the incident sunlight, it will radiate some of this heat out in the form of infrared radiation. The atmosphere will absorb some of it, while clouds can reflect some of the radiation back to the terrain. Some of the radiation will escape to outer space, which we model using equation 4.5.

For simplicity, we do not model cloud reflection and instead assume that a general proportion R of the energy radiated out is absorbed. This proportion can be tuned. For a terrain cell i, we model the change in temperature through radiation absorbed by the air voxel i above it using equation 4.1 again. The change in temperature for the air voxel uses a variant of the same equation, with the specific heat capacity (c) and density (ρ) changed to be that of air at the given altitude, times the constant factor R.

$$\Delta T_A = \frac{\Delta Q_i\ \Delta t}{c_{\text{air}}\ \rho_{\text{air}}} R \tag{4.6}$$

ΔQ_i will match in both equations, and is calculated as follows:

$$\Delta Q_i = \epsilon_i \sigma \left({T_T}^4 - {T_A}^4\right) \tag{4.7}$$

[1]Given the limited time domain, Z is fixed for all cells for the duration of a simulation.

where ϵ_i is the emissivity of the terrain in cell i, σ is the Stefan-Boltzmann constant, and T_T and T_A is the temperature of the terrain in cell i and the temperature of the air in voxel i respectively.

4.2.3 Advection

To model the movement of wind and the quantities carried by it, we implement a fluid dynamic simulation for each horizontal layer, using the equations described in section 2.4. We iterate through each horizontal layer, and through each cell in the layer. Each cell has a number of quantities that need to be advected, listed in table 4.6.

For each point $\vec{x}_i$ where values are stored (scalar values at the centre of the cell, with the velocity separated and stored on the edges of the cell), we calculate a virtual particle. Tracing the virtual particle to its position in the previous timestep, we sample new values for the values stored at $\vec{x}_i$. By using a memory buffer, the cells of the layer can be iterated through in parallel, which provides a significant performance increase.

Name	Symbol	Unit
2D velocity	$\vec{U}$	$m \cdot s^{-1}$
Temperature	T	K
Moisture	M	$kg \cdot m^{-3}$
Condensed water	W	$kg \cdot m^{-3}$
Particulate density	ρ	$kg \cdot m^{-3}$

Table 4.6: The list of quantities that are advected by each 2D layer.

4.2.4 Terrain friction

On Earth, the movement of the air is slowed by friction with the terrain. Since our fluid model is limited in resolution through the use of layers, an adjustment is necessary to model this friction.

We reuse the system proposed by Vimont et al.[Vim+20], with slight adjustments to suit our grid system (see section 2.4.5). An orographic damping factor σ_i is calculated for each point i where velocity is stored in the air layer grids. The elevation at point p directly below point i ($h(p)$) is sampled and interpolated if not exactly on a value. Since we utilise an MAC grid, velocity values are stored on the edges of a cell, while elevation values are stored (like all scalar values) at the centre of a cell.

σ_i is then calculated as follows:

$$\sigma_i = 1 - min\left(1, max\left(0, \frac{h(p) - h_i}{\Theta_i}\right)\right), \tag{4.8}$$

where h_i is the height of the air layer, and Θ_i is the thickness of the air layer. This damping factor is then applied to the velocity at point i. This damping leads

to non-convective vertical motion, more accurately reflecting real life despite the limited resolution. Such orographic lift effects include stacks of lenticular clouds that form behind a mountain peak, or the long, flat clouds that form on the peaks of long mountains, such as Table Mountain. See section 4.3.2 for a more detailed explanation of this.

4.2.5 Pressure correction

After the various quantities in the horizontal layers have been advected and the velocities dampened, the velocity vector field $\vec{U}$ is no longer divergence-free. To conform to the Navier-Stokes equations, a correction must be applied. Following the method described in section 2.4.5, we iterate through each cell in a horizontal layer and calculate the divergence for that cell, as well as the relevant coefficients for the sparse linear system.

Once all cells have been iterated through, we calculate the pressure correction terms (one per cell) using the conjugate gradient solver from the Eigen package[GJ+10]. This solver was chosen for its ease of use and high performance, being highly parallelised. These pressures are used to calculate the corrections to the velocity field as per equations 2.10 and 2.11. Each horizontal layer in the model is iterated through sequentially.

4.2.6 Phase transition

A key component of the system is to accurately model the phase transitions of the moisture in the air and the effect this has on the temperature and appearance of the skyscape. Given the extreme conditions present in a volcanic plume, additional care is required here compared to traditional solutions.

Water should evaporate if the saturation vapour pressure is greater than the partial pressure exerted by the moisture in the air. Conversely, if the saturation vapour pressure is less than the partial pressure exerted by the moisture in the air, water should condense. This phase exchange is a dynamic equilibrium that depends largely on temperature.

Within the plume, temperatures can exceed the *critical temperature* of water, which is $647K$. This is the temperature at which water cannot remain a liquid, regardless of the pressure. In this state, we force the liquid in a cell to immediately transition to a gaseous state, with no moderation dependent on the partial pressure exerted by the moisture in the air.

We iterate through each cell in a layer in parallel, and each air layer in turn. For each layer, we calculate the ambient pressure at the air layer's height P_h using the barometric formula[Atm76]:

$$P_h = P_b\left(1 - \frac{L}{T_b}h\right)^{\frac{gM}{R^*L}} \tag{4.9}$$

P_b refers to the base pressure at sea level (which is 101 325 pascal on Earth), L refers to the lapse rate (the change in temperature as a function of altitude), T is the average temperature at sea level (288.15 K on Earth), and h is the height of the air layer in question.

In the exponent, g is the gravitational acceleration (9.8 $m \cdot s^{-2}$ on Earth), M is the molar mass of air (0.029 $kg \cdot$ mole on Earth), R^* is the universal gas constant, and L is once again the lapse rate.

It must be noted that the pressure values generated by the advection system are dimensionless and not physical. Therefore, those values cannot be used to calculate the ambient pressure in a cell.

Instead, for each cell, we sample the temperature, the moisture, and the liquid water content of the cell. Cells that are marked as solid cells (i.e., the terrain impinges on the air layer in that cell) are ignored. We then calculate the saturation vapour pressure for the given temperature using the Arden-Buck equation (equation 4.10[Buc96]), where T is the cell temperature in degrees Celsius.

$$\begin{aligned} P_s(T) &= 6.1121\ exp\left(\left(18.678 - \frac{T}{234.5}\right)\left(\frac{T}{257.14 + T}\right)\right) \quad \text{if } T \geqslant 0\ ^\circ C \\ P_s(T) &= 6.1115\ exp\left(\left(23.036 - \frac{T}{333.7}\right)\left(\frac{T}{279.82 + T}\right)\right) \quad \text{if } T < 0\ ^\circ C \end{aligned} \tag{4.10}$$

This is a change from the system used by Vimont et al.[Vim+20], who used Tetens' equation[Tet30] to calculate the saturation vapour pressure. This change is driven by the higher temperatures encountered in the plume and its immediate vicinity – the Arden-Buck equations prove more accurate than Tetens' equation over a wider range of temperatures when compared to the measured values of the saturation vapour pressure of water[RLB19].

We calculate the density of the air ρ in the cell using the ideal gas law (equation 2.1 in a molar form:

$$\rho = \frac{P_h M}{R^* T_b \left(1 - \frac{Lh}{T_b}\right)} \tag{4.11}$$

Here, P_h refers to the pressure at altitude h, as calculated by equation 4.9. M is the molar air mass, R^* is the universal gas constant, T_b is the average temperature on the planet, and L is the lapse rate.

The density is used to calculate the absolute humidity H, a simple ratio between the moisture m and the density ρ. The absolute humidity is then used to calculate the partial pressure exerted by the moisture (P_m) in the cell, where P is the atmospheric pressure in the cell:

$$P_m = P\frac{H}{0.622 + H} \tag{4.12}$$

Similarly, the saturation humidity, i.e., the humidity that would be present in a cell in equilibrium, is calculated using:

$$H_s = \frac{0.622 + P_s}{P - P_s} \tag{4.13}$$

At this point, there are three possible options in the cell. The liquid water can evaporate, the liquid water can boil, or the moisture can condense. Depending on the atmospheric pressure, the partial pressure exerted by the moisture, and the saturation vapour pressure, we determine which of these options should occur.

If the saturation vapour pressure P_s is greater than the atmospheric pressure P, all the liquid water in the cell will *boil* and be converted to moisture. If the partial pressure P_m is less than the saturation vapour pressure, the liquid water will partially *evaporate*. Lastly, if the partial pressure is greater than the saturation vapour pressure, the moisture will partially *condense* into liquid water.

For evaporation and condensation, the difference ΔH between the saturation humidity H_s and the absolute humidity H is calculated and used to determine the mass of water that changes phase. This is multiplied by a user-determined parameter J for more realistic behaviour. In our case, J is set to 0.1 for both evaporation and condensation. This parameter is required given that our approximation is a discretisation of a continuous process with a relatively large time step. Without this modulation, a cell can oscillate between mostly gaseous vapour and mostly liquid moisture at each time step, which is unrealistic and does not approach the equilibrium that is observed in nature.

The final part of the phase transition engine is the heat exchange component. To change phase from a liquid to a gas, energy is required, and this energy is subtracted from the temperature in the cell. Similarly, to change phase from a gas to a liquid, energy must be removed from the gas, which is added to the temperature in the cell.

Specifically, the change in temperature is calculated as follows:

$$\Delta T = \frac{L_v \Delta H}{c_A \rho}, \tag{4.14}$$

where L_v is the enthalpy of vaporisation for water ($2257\ J \cdot g^{-1}$), and c_A is the specific heat capacity of dry air ($1.0035\ J \cdot g^{-1}$). This ΔT is then added to the temperature of the cell.

4.2.7 Moisture uptake

Similar to the phase transition mechanics, the atmosphere is capable of taking up moisture from the terrain below, or condensing out onto the terrain if it is saturated.

Specifically, if the partial pressure exerted by the moisture in the air is less than the saturation vapour pressure, a portion of the liquid water in the terrain can evaporate and be added to the atmosphere. Inversely, if the partial pressure exerted by the moisture in the atmosphere exceeds the saturation vapour pressure, the moisture

can condense onto the terrain.

For each terrain cell, we sample the air cell directly above it. We use equations 4.9, 4.10, and 4.11 to calculate the ambient pressure, the saturation vapour pressure, and the density of the air in the air cell. Similarly, equations 4.12 and 4.13 are used to calculate the partial pressure exerted by the moisture in the atmosphere and the saturation humidity.

As in the phase transition engine, the saturation vapour pressure and partial pressure of the moisture in the air are compared to determine if evaporation or condensation should take place. The change in temperature resulting from this phase transition is calculated using equation 4.14, with the specific heat capacity changed to be that of the terrain instead of the air.

4.3 Interlayer communication

While the previous sections set out how we modelled the atmosphere within the horizontal layers, real atmospheric phenomena are rarely limited to single layers. Therefore, we need to model the transfer of quantities such as temperature or moisture between different layers. Several phenomena require interaction between layers or extend across multiple layers, such as convection or orographic lift. Convection is a major component in cloud formation and cloud types, along with being a key component of the chain of events that lead to wind forming. In particular, we focus on the modelling of precipitation, as a one-way phenomena from higher layers to lower, and the bidirectional exchange of quantities due to temperature and pressure differences.

4.3.1 Precipitation

Precipitation refers to the products of condensed moisture in the atmosphere falling from the clouds due to gravity. In our case, we focus only on two forms of precipitation: water, known as rain, and volcanic ash mixed with water, referred to as ash rain.

We model precipitation using a physical approximation of coalescence (or aggregation, when referring to ash rain). Small droplets coalesce together to form bigger droplets, until the droplets grow large enough to overcome air resistance and electrostatic forces keeping the cloud together.

This method contrasts with the probabilistic models common in meteorology [Søn+20; Rav+21]. This is needed to accurately transfer physical quantities between the discrete layers using our procedural approach to inter-layer exchange (see section 4.3.2).

We use the same method for both rainfall (water droplets) and ash rain (fine ash mixed with water). For each cell in the system, we sample the horizontal and vertical velocities in the cell. We separate horizontal F_{hor} and vertical F_{ver} scaling factors, since the vertical velocity (uz) is a procedural rather than physical value.

Parameter	Rainfall	Ash rain
F_{hor}	0.5	0.5
F_{ver}	0.1	0.1
Ω_{base}	2	0.02
F_L	0.5	0.001

Table 4.7: A list of the scaling factors used for the two precipitation modes.

The magnitude of this velocity is used as an approximation for turbulence, which influences the collision rate and therefore the total precipitation.

We calculate potential precipitation ρ_{pot} using the current density ρ_i of precipitable matter in the cell scaled by a coalescence factor C_F, according to:

$$\rho_{\text{pot}} = \rho_i \times C_F, \tag{4.15}$$

$$C_F = \min(|\vec{H}_i| \times F_{\text{hor}} + |uz_i| \times F_{\text{ver}},\ 1.0). \tag{4.16}$$

If this potential precipitation exceeds a calculated resistance Ω_{fall}, a portion of the precipitation $\Delta\rho$ falls to the layer below, with scaling factors from the timestep (Δt) and the height between the layers (h) to prevent instantaneous transport:

$$\Omega_{\text{fall}} = \Omega_{\text{base}} + uz_i \times F_L \tag{4.17}$$

$$\Delta\rho = \begin{cases} \rho_{\text{pot}} \cdot F_{\text{base}} \cdot \Delta t \times \frac{1}{h} & \text{if } \rho_{\text{pot}} > \Omega_{\text{fall}} \\ 0 & \text{otherwise} \end{cases} \tag{4.18}$$

If the precipitation reaches the ground (i.e., the cell below is impinged on by terrain), it is removed from the atmospheric system and added to the terrain. To differentiate between rainfall and ash rain, the scaling factors F are changed to suit the specific precipitation characteristics. Table 4.7 lists the values we used in practice for the various scaling factors.

4.3.2 Layer exchange

The bidirectional exchange of quantities between layers is divided into two components: the convective (driven by buoyancy and density differences) and non-convective (driven by pressure differences between layers). In practice there is a minor effect from the Coriolis effect as well, with westward moving air rising and eastward moving air descending. However, this occurs on massive scales, many hundreds of kilometres across, far greater than we consider in this work. Therefore, we focus only on the convective and pressure-driven vertical motion.

In a full 3D fluid solver, this movement would arise naturally from the solutions to the fluid equations. Since we have discrete horizontal layers, we must explicitly calculate the vertical velocities and the attendant consequences. For performance, we do this in a procedural manner.

Convective

The majority of vertical motion in the atmosphere is convective in nature, driven by the differing densities between hot and cold air.

For each cell in a vertical layer, we sample the cells above and below for temperatures (T_A, T_B), moisture (M_A, M_B), and altitude (H_A, H_B). We calculate the barometric pressure for the layers above and below (P_A, P_B) using equation 4.9. Using the pressure and temperature, we calculate the quantity of air per volume in the cell:

$$N_l = \frac{P_l}{R^* T_l}, \tag{4.19}$$

where l refers to a specific layer, either above or below the vertical exchange layer, and R^* is the universal gas constant.

These values are then used with the moisture values sampled to calculate the specific humidities:

$$Q_l = \frac{M_l}{N_l} \tag{4.20}$$

This brings us to the moist static energy equation, which describes the rough amount of enthalpy present in an air parcel[WH06].

$$MSE_l = c_A \cdot T_l + g \cdot H_B + L_v \cdot Q_l \tag{4.21}$$

c_A refers to the specific heat capacity of air, g is the acceleration due to gravity, and L_v is the latent heat of vaporisation of water moisture. The difference between the MSE of the layers above and below gives an approximation of the vertical velocity between the two layers in that cell.

Therefore, the vertical velocity uz_i is calculated as the difference between the MSEs of the layers, multiplied by a scaling factor C_{cv}:

$$uz_i = C_{cv}(MSE_B - MSE_T) \tag{4.22}$$

The scaling factor provides user control over how energetic the atmosphere is, with greater values leading to more convection.

Vertical movement will occur from the area with higher enthalpy to the area with lower enthalpy. This is a dynamic equilibrium much like the phase transition. The velocity calculated is not preserved through multiple simulation steps, instead being calculated anew every step.

Non-convective

While the pressure values calculated by the pressure correction system are non-physical and unitless, they are equivalent between different layers. Therefore, we can calculate a vertical velocity induced by a pressure difference between layers.

For example, if a layer is impinged on by terrain such as a hill or mountain, a high pressure area forms at the windward side. This high pressure zone leads to orographic lift since the layer above does not have the same obstacle and thus a lower relative pressure.

For each cell in the convection layer, we sample the pressure values in the horizontal layers above and below. We normalise these values to ensure that the relation between the layers can be established. We calculate the change in vertical velocity (from the initial velocity calculated by equation 4.22) due to pressure differences (Δuz_{iP}) as the difference between the normalised pressure in the cell above ($\widetilde{P}_{i_T}$) and in the cell below ($\widetilde{P}_{i_B}$) multiplied by a scaling factor C_P.

$$\Delta uz_{iP} = (\widetilde{P}_{i_T} - \widetilde{P}_{i_B}) \times C_P \tag{4.23}$$

Smoothing

Since the vertical motion is a continuous process in a dynamic equilibrium, the discretisation we apply to simulate it can result in oscillations forming across time steps with sharp discontinuities between neighbouring cells. To dampen this effect, we apply a 2D moving window average smoothing to the vertical velocity field in each layer. The moving window has a radius of 3 cells. This size is chosen as a compromise between performance and effectiveness, given the computation requirements of $O(n)$ for n cells. Since the distance between discontinuities is small, a small window size suffices in this case. The simple moving average was chosen for its ease of implementation and adequately smoothed out the discontinuities.

Vertical transport

Once the vertical velocity has been calculated, we transport scalar values between the horizontal layers. For a scalar S in cell i, we transport the difference between the scalar values (see table 4.1) of the above and below cells multiplied by a coefficient that is scaled by the simulation timestep and the height between two layers.

We calculate the coefficient according to the following equation:

$$C_i = 1 - \exp\left(\frac{-|uz_i|\Delta t}{H_A - H_B}\right) \tag{4.24}$$

The amount of the scalar values transported is then calculated (for each scalar quantity S) as:

$$\Delta S = C_i \times (|S_B - S_A|) \tag{4.25}$$

ΔS is added to the destination layer in the direction of the vertical velocity, and subtracted from the source.

Inflow and outflow approximation

The final component of the vertical exchange system is to model the inflow and outflow of air in convective columns. Rising air will result in a low pressure area directly below it, and a high pressure area above it.

We model this per cell in the horizontal air layers. For each cell, we sample the vertical velocity immediately above and below the cell. We calculate the net flow into the cell by subtracting these values from each other:

$$uz_{\text{net}} = uz_B - uz_A \tag{4.26}$$

With this, we then calculate a change in horizontal velocity (Δu_x and Δu_y) using the net flow and a scaling factor.

$$\Delta u = C_u \times uz_{\text{net}} \tag{4.27}$$

If the net flow into the cell is positive, each velocity value on the cell (see section 2.4.5) will be modified by Δu for a greater flow out of the cell. Similarly, if the net flow is negative, each velocity value will be modified by Δu for less flow out of the cell.

It must be noted that these modifications affect the 2D velocities of the horizontal layers, and thus the velocity fields are no longer divergence free (violating equation 2.4). The advection scheme we use is more sensitive to instabilities than the one used by Vimont et al.[Vim+20]. This sensitivity is overcome by running the vertical coupling step at a lower frequency than the advection step (see algorithm 1). If the vertical coupling step was run at the same frequency, the velocity fields would never be in a divergence-free state. By allowing a few steps where the velocity fields are divergence free, the inherent stability of the system (as described in section 2.4.4) can be maintained, albeit with less guarantees than if the vertical coupling was integrated as part of the pressure solve. In practice, we found that a ratio of one vertical coupling step to three advection steps was enough for indefinite stability.

Algorithm 1 The steps for a full time step of the complete weather subsystem

```
N ← MaxSteps                          ▷ Maximum number of steps can be
x ← 0                                 ▷ time bound or it can be set by the user.
K ← CellCount                         ▷ Cell count is the number of cells in the grid
Δt ← TimeStep                         ▷ The timestep used for most engines
Δt_VCE ← VCETimeStep                  ▷ The timestep used for the vertical coupling engine
while x < N do
    Step terrain heating engine by Δt                          ▷ Section 4.2.1
    Step atmospheric heating engine by Δt                      ▷ Section 4.2.2
    Step moisture uptake engine by Δt                          ▷ Section 4.2.7
    Step precipitation engine by Δt                            ▷ Section 4.3.1
    Step advection engine by Δt                                ▷ Section 4.2.3
    Step obstacle friction engine by Δt                        ▷ Section 4.2.4
    Step pressure correction engine by Δt                      ▷ Section 4.2.5
    if x × Δt mod Δt_VCE = 0 then
        Step vertical coupling engine by Δt_VCE                ▷ Section 4.3.2
    end if
    Step phase transition engine by Δt                         ▷ Section 4.2.6
    x ← x + 1
end while
```

Chapter 5

Plume Modelling

To produce a visually realistic plume, we build on the proxy-geometry approach proposed by Lastic et al.[Las+22]. As mentioned in chapter 3, their approach assumes the atmosphere to be a static boundary condition. In this chapter, we provide an overview of Lastic et al.'s method, set out the changes necessary to couple this model with the atmospheric model described in the previous chapter, and describe the additions made to represent some of the visual phenomena present in volcanic eruptions.

5.1 Plume model

We only provide an overview of the model proposed by Lastic et al., and refer the reader to the original paper[Las+22] for a full account.

5.1.1 Plume slices

The core dynamics of the eruption are modelled using cylindrical slices emitted from the vent. Each slice carries attributes for position, size, temperature, ash density, and moisture. Each slice is modelled as a Lagrangian particle, with buoyancy, friction, gravity, and wind forces acting upon it. Equation 5.1 is used to compute the acceleration applied to a given slice, where Δt is the timestep and M is the mass of the slice. $\vec{F_b}$ is the buoyancy force, $\vec{F_f}$ is the friction force, $\vec{F_g}$ is the gravitational force, and $\vec{F_w}$ is the force applied by the local wind. Figure 5.1 depicts the slice and sphere model we use.

$$\Delta\vec{v} = \frac{\Delta t}{M}\left(\vec{F_b} + \vec{F_f} + \vec{F_g} + \vec{F_w}\right) \tag{5.1}$$

In addition, we model the entrainment of the ambient air into the eruptive column. This entrainment of ambient air is what causes the plume to rise (due to a reduction in density as the entrained air heats up) and is the origin of many of the visual features of the plume, such as the billowing appearance. Each slice includes a volume of the surrounding air at each timestep. The mass of air entrained is dependent on the velocity of the slice and the direction of the local wind. The entrained air changes the temperature of the slice, since the ambient air is at a lower temperature than the plume. The slice also decreases in density, to model the sedimentation of

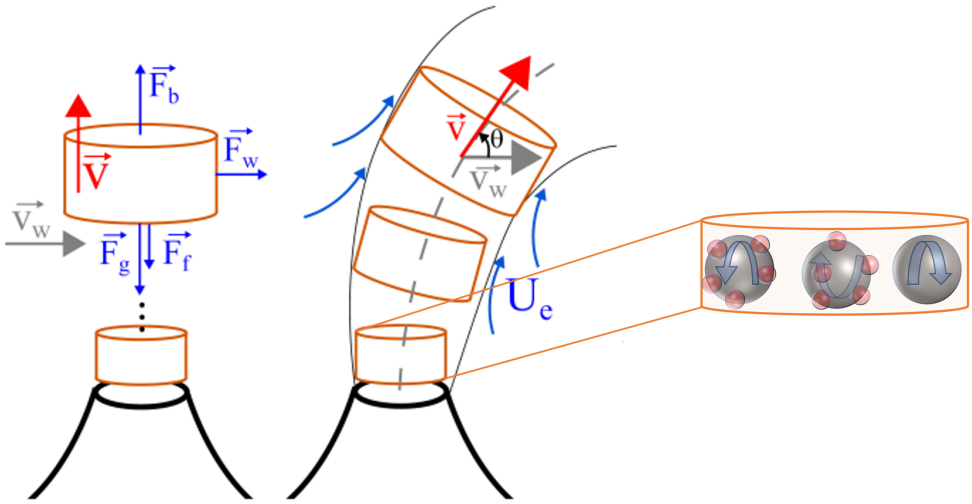


Figure 5.1: The plume model is divided into slices, treated as Lagrangian particles, and a hierarchy of spheres that provide lower level detail. On the left, the slices and the physical dynamics influencing it are shown, along with the buoyancy ($\vec{F}_b$), friction($\vec{F}_f$), gravitational ($\vec{F}_g$), and wind ($\vec{F}_w$) forces. On the right, a slice with attached rotating spheres (number limited for legibility) is shown. The grey spheres are the primary spheres, while the red spheres are at the second level of the hierarchy. The spheres detach from the slice after the slice reaches neutral buoyancy.

ash and tephra. Using the ideal gas law (equation 2.1), we can calculate the new mass of the slice.

5.1.2 Plume spheres

To model the local velocity in the plume (as compared to the global velocity of the plume as a whole), procedural rotating spheres of multiple sizes are used. This is in accordance with the superficial appearance of a billowing plume, with rotation from the core of the plume to the outside due to the convective column.

The spheres at the largest scale are of a similar radius to the slice at that altitude. In the model, spheres are placed on a slice, with their centres at a distance from the slice centre equal to $r/2$, where r is the radius of the slice. The spheres are spaced to ensure overlap between each other, with minor noise perturbations to the position and size for natural variation. The spheres move with the slice, and the rotation speed matches the speed of the air moving past on the outside of the column.

Additionally, subspheres are placed randomly across the surface of each sphere. These subspheres are set to be one fifth the size of the sphere to which they are attached to, and their motion is locked to that of the parent sphere.

Once the plume reaches neutral buoyancy, it will spread out laterally in response to the high pressure zone created by the convective column. To accommodate this different visual appearance, the large spheres are disconnected from their parent slices once the slice reaches its peak height and allowed to spread out, advected by the local wind field.

5.2 Sphere Enhancements

We make a number of changes compared to the original approach[Las+22] in order to accurately portray lightning and ash rain. In addition to only slices carrying temperature, ash density, and moisture values, each sphere now tracks ash and electrical charge. This is required for accurately transferring the ash to the atmosphere once detached from the parent slices. At each timestep, the values for ash, moisture, and temperature are updated by the slice and copied to the attached spheres.

The spheres and subspheres are treated differently when interacting with the atmosphere (see chapter 6). Temperature and moisture are transferred from the primary spheres to the atmosphere, while the subspheres are responsible for transferring ash to the atmosphere. This reflects the accuracy requirements, where the ash must be transferred with as much spatial precision as possible, while the moisture and temperature do not need the same level of precision.

During the umbrella phase, the plume becomes thinner vertically and elongates as it moves away from the vent. This is a result of the plume umbrella behaving like a gravity current[WK94]. This phenomena is replicated in our system by having the spheres gradually shrink in the vertical direction once detached from the slices, becoming ellipsoids. This shrinking is done as a function of time and wind speed, with more time and greater wind speed leading to a faster rate of compression.

5.3 Lightning

As one of the most striking aspects of volcanic plumes of all types and sizes, volcanic lightning has been extensively studied both in nature and in laboratory conditions[CG22]. Indeed, so prevalent is volcanic lightning across an extremely broad range of eruption types, that it is considered "an intrinsic property of volcanic ash plumes[CG22]." Lightning in any cloud system is an electrostatic discharge between two regions that are electrically charged, with the lightning temporarily neutralising these regions. To accurately mimic lightning, the mechanisms that result in electrically charged regions of the air must be explored.

While it is still not fully understood, electrification in water clouds is thought to be the result of water droplets and different forms of ice crystals colliding with each other under the strong convective turbulence in a cloud. The different types of droplets and crystals have different shapes and sizes and therefore move on different trajectories, leading to the now charged crystals accumulating in different parts of the cloud.

Several hypotheses have been accepted as likely mechanisms for electrification in volcanic plumes, and we focus on three of these in our work. First, *fracto-electrification* takes place in or near the vent as tephra forms. Tephra is formed when hardened magma or lava is shattered by the eruption. Many small fragments form and exert friction on each other, which leads to static charge formation. This mechanism is approximated by giving each slice and its child spheres an initial, constant charge as it is emitted from the vent.

In the jet and convective phases, friction between the different components of the eruption column leads to a build-up of charge in *tribo-electrification*. These components have different chemical compositions and structures, such as glass (amorphous crystal solids) or rocks (heterogeneous aggregates of minerals). They also have different granularities, referring to the density, shape, and size. It must be noted that laboratory experiments have demonstrated that moisture has a significant suppressing effect on this kind of charge build-up[Ste+19; Mén+20]. We therefore add an exponential decay based on the ambient moisture below the -20 °C isotherm[1]. At temperatures below -20 °C, very little moisture remains in the atmosphere.

At such higher altitudes, supercooled water droplets and ice flakes collide and experience friction, similar to electrification in thunderstorms. This is known as *water and hydro-meteor electrification*. However, due to the presence of ash particles that serve as crystal seeds, substantially more ice can form in a plume compared to a thunderstorm of equal temperature. Therefore, the build-up of charge is enhanced in the volcano plume. This method of electrification is considered to be considerably stronger than tribo-electrification[CG22], with roughly an order of magnitude more charge.

To accommodate these different sources of electrification, we have each sphere p track its own electrical charge, calculated from the sphere's centre. This is a dimensionless value L_p that is increased by ΔL_p at each timestep of the plume system. We calculate ΔL_p using equation 5.2:

$$\Delta L_p = \begin{cases} k_1 \cdot \overline{A_p}(k_{hA}) \cdot \overline{\|\mathbf{V}_p\|}(k_{hV}) \cdot e^{-10\overline{M_p}(k_M)} & \text{if } T > 253.15\ K \\ k_2 \cdot \overline{W_p}(k_W) \cdot \overline{H_p + U_p}(k_{cV}) & \text{otherwise.} \end{cases} \tag{5.2}$$

Note that this distinguishes between the two electrification regimes below and above the -20 °C ($253.15\ K$) isotherm. At temperatures above the isotherm, we include terms for ash density, sphere velocity, and the exponential decrease with moisture content mentionied earlier. At temperatures below the isotherm, we take into account the atmospheric velocity and the amount of liquid water in the surrounding atmosphere.

The bar function used in equation 5.2 denotes a capped normalisation, where for a constant f, $\overline{f}(k) = \min(f, k)/k$. In practice, we use the values listed in table 5.1 for capping the various electrification effects.

The actual electrical discharge in the form of lightning is handled by the procedural amplification and offline rendering system. See section 6.2.4.

5.4 Embedded Clouds

Our atmospheric system consists of multiple horizontal layers, with a sparse vertical distribution. This is designed to capture the layered nature of the atmosphere in

[1]This isotherm is found at 4117 m above sea level in our simulation, though its height varies in nature depending on the atmospheric composition.

Cap	k_1	k_2	k_{hA}	k_{cA}	k_{hV}	k_{cV}	k_M	k_W
Value	0.1	0.5	100	5	400	20	50	50

Table 5.1: The values used to cap various effects during the calculation of ΔL_p for each sphere p.

typical weather conditions. However, this is too coarse a resolution (especially vertically) to accurately model the air parcels in and immediately around the plume. The air in or near the plume can be subject to significant uplift and extreme turbulence leading to distortion.

The various embedded clouds described in section 2.3.1 are a consequence of the fine scale behaviour of the plume. To mimic these features, we create deformable sheet surfaces that are sampled vertex-wise. This sampling is done at a much finer resolution than the atmospheric system, particularly in the vertical. Where the atmosphere system can have layers that are up to 2500 metres in vertical extent, the sheets are only 100 metres thick. The number of sheets can be varied by the user, but in practice, we found that three sheets are sufficient to replicate the appearance of the skirt and cap clouds found in nature.

A sheet is created as a horizontal disc, centred above the volcano vent, and at a specific altitude. Prior to the eruption, multiple sheets are created and spaced vertically to represent naturally occurring layers of alternating moist and dry air. Each vertex σ on the sheet is initialised with moisture M_σ and liquid water W_σ, both of which are trilinearly interpolated from the nearest grid cells in the atmospheric system.

To model vertical uplift, for each vertex σ, we sample the vertical velocity of all slices and spheres in the plume system at each timestep. For each slice and sphere p with an altitude h_p below h_σ, a weighted vertical velocity $V_p \cdot \Psi(V_p)$ is added to the velocity of vertex σ. The weighting function $\Psi(V)$ is dependent on the distance d between the sphere p and the vertex s, and is calculated according to equation 5.3:

$$\Psi(V) = \begin{cases} 1 & \text{if } \|V\| <= r_s \\ S_1(1-d) & \text{if } \|V\| > r_s \ \&\ d = \|(\vec{V}_x/r_h, \vec{V}_y/r_h, \vec{V}_z/r_v)\| < 1 \\ 0 & \text{otherwise,} \end{cases} \tag{5.3}$$

The smoothstep function is defined as $S_1(x) = 3x^2 - 2x^3$, while $\vec{V}$ is the vector from the surface of sphere p to the sheet vertex σ. If the sample is within the sphere's inner radius r_s, then the weight is set to unity. Outside of the inner radius, the weight tails off according to the smoothstep function on the distance d. The distance is calculated as the normalised distance within a horizontal offset region r_h and a vertical offset region r_v. Outside of this ellipsoid, the weight is 0. Figure 5.2 depicts the uplift force applied to vertices, as a 2D slice through the centre of the skirt.

By varying the size of the outer ellipsoid (r_h, r_v), we can procedurally model the differences between air parcels pushed ahead of the plume compared to parcels

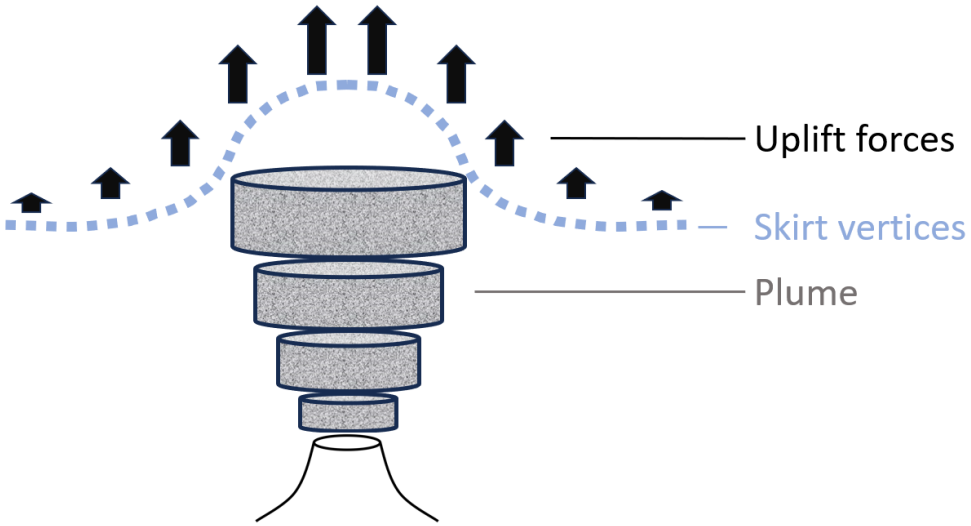


Figure 5.2: The uplift force on each vertex depends on the distance from the slices and spheres of the plume. The further away from the plume the vertex is, the less the force that is applied.

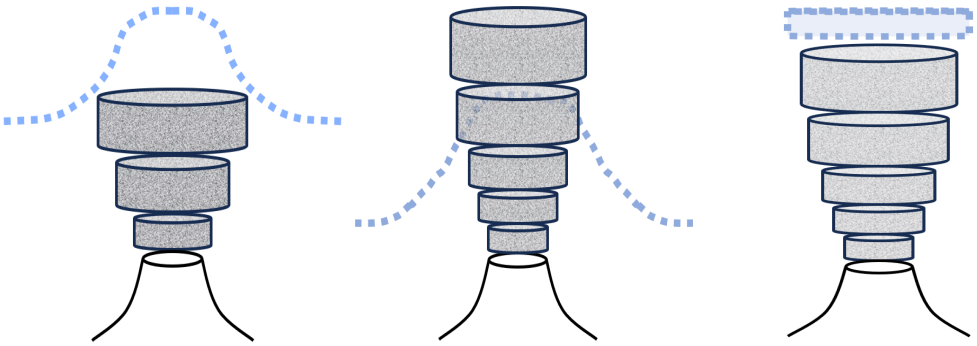
that are dragged along by the plume. This approximates the deformability of the embedded cloud, where cap clouds are less deformable than bell or skirt clouds. We continue to model the uplift until the sheet reaches a preset altitude where it is neutrally buoyant.

As the sheet vertices are uplifted, adiabatic cooling is applied, using exactly the same mechanics as the atmospheric system for condensation or evaporation (see section 4.2.6). At higher altitude, the sheet vertices are usually at a lower temperature, with temperature determined by the ambient temperature at the vertex's location, sampled from the atmospheric system. If sufficiently cooled given the moisture values of the vertex, some moisture condenses to add to the liquid water content. If the liquid water is at sufficient concentration, the sample becomes visible as a cloud. The deformation of the sheets and subsequent shape give the characteristic appearance of bell, skirt, and cap clouds. Figure 5.3 depicts the three types of embedded clouds that can be modelled by our system.

5.5 Transient condensation clouds

In explosions, such as bomb blasts and volcanic eruptions, the initial shock wave can be powerful enough to cause transient condensation clouds to form in humid conditions as the pressure wave propagates. These clouds are formed by an adiabatic drop in air temperature as the pressure wave passes. This causes the moisture in the humid air to condense out and form a cloud of droplets. After a few seconds the air temperature returns to normal and the droplets evaporate again.

One way to model this phenomenon is to directly decrease and then restore pressure in the atmospheric system for the affected cells. However, as mentioned before, the pressure values we have are dimensionless and can therefore not be used in the phase transition updates. The values calculated by the advection system are



(1) Bell (2) Skirt (3) Cap

Figure 5.3: The three variants of embedded clouds that we simulate in our framework. [1] Bell clouds form as moist air layers are lifted more in the centre of the plume and condenses out. [2] Skirt clouds are the result of the plume breaking through bell clouds and obscure the central bell. [3] Cap clouds are confined to immediately above the plume, with no layers visible around the plume as is the case for bell and skirt clouds.

purely relative, with an actual magnitude dependent on the velocities of the system. These values are not representative of the actual values found in the atmosphere, nor does it have a comparable range to values found in the atmosphere. In contrast, all pressure values used in calculating phase transitions require actual pressure values in the range found in the atmosphere.

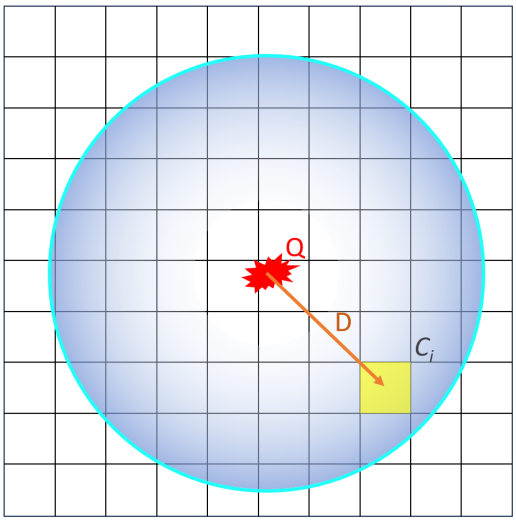


Figure 5.4: A pressure wave originating at Q (the red star) propagates outwards (represented by the cyan circle). For each cell C_i within the pressure wave, we calculate the distance D_i between C_i and Q. The temperature for C_i is reduced by N/D_i^2 to calculate T_i^*, with T_i^* increasing over time until it is restored to T_i 10 seconds after the pressure wave passes by. Darker blue indicates lower values for T^*, while white indicates ambient temperature values (T).

Instead, we temporarily reduce the temperature used in the phase transition equations (equation 4.10), again using the barometric air pressure calculated by equation 4.9. We simulate a pressure wave centred on the volcano vent Q with an

initial non-dimensional strength N. Once the wave is initiated (timing controlled by the scenario), it propagates outwards at the speed of sound. For each cell C_i that is in the spherical area encompassed by the pressure wave, we calculate the distance D_i between C_i and Q.

The pressure wave strength decays as the square of its radius. The adjusted temperature T_i^* is calculated by subtracting a value of N/D_i^2 multiplied by a scaling factor to fit the temperature scale. The adjusted temperature decays as time passes, and is removed completely 10 seconds after the wave first passes cell C_i, thereafter conditions are allowed to return to the normal dynamic equilibrium.

Chapter 6

Combining submodels and amplification

To accurately model the volcanic plume, a model of the atmosphere is required to capture the influence the ambient atmosphere can have on the appearance of a volcanic plume, such as embedded clouds (section 5.4) or ash rain (section 4.3.1). Lastic et al.'s original plume model used a static boundary condition, which is not accurate to reality. Similarly, many of the atmospheric characteristics present near a volcanic plume require a model of the plume to be known.

If we set our system's parameters for maximal simulation accuracy, the computation cost would be prohibitive. Therefore, we optimise and simplify as many systems as possible to achieve the best performance possible while balancing accuracy. The atmospheric system operates on a coarse scale, which does not result in a realistic appearance of the clouds from close range. Similarly, the plume, while comparatively fine-scaled, is also at too coarse a resolution to accurately capture the appearance of a volcanic plume in nature. By adding noise to the appearance dictated by the simulation models, we can achieve a significantly more realistic appearance at very little cost.

In this chapter, we describe how we combine the atmospheric and volcanic plume submodels, as well as describing the procedural amplification applied to generate the final visual results. As mentioned in section 5.3, the discharge of the electric charge is handled solely by this system and is therefore described in this chapter as well.

6.1 Model combination

In order to accurately and effectively couple the two submodels, We need to transfer physical quantities such as ash, moisture, temperature, and wind between the two submodels. Without such transfer, many processes found in nature such as entrainment or ash rain cannot be accurately modelled.

We encounter two major challenges in the combination. First, the two submodels are run at different frequencies. The volcanic plume requires smaller timesteps due to the higher velocities encountered in the eruption column. In our framework, we run the submodels at a ratio of 50 to 150 steps of the plume system ($\Delta t = 0.02$ ms)

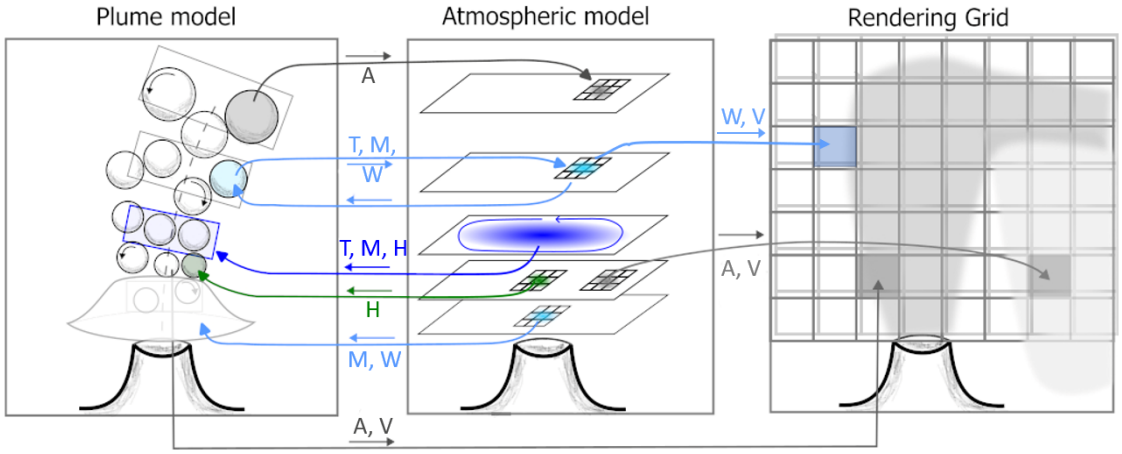


Figure 6.1: We transfer physical quantities between the two submodels, as well as the final rendering system. We transfer ash A, moisture density M, water content W, temperature T, 3D velocity V, and horizontal 2D velocity H.

for each step of the atmospheric system ($\Delta t = 1 - 3$ ms). Immediately before each step of the atmospheric system, the slower of the two, we synchronise and transfer the physical quantities from the plume to the atmosphere. Immediately after, before the next plume step, we transfer the physical quantities from the atmosphere to the plume.

The second significant challenge deals with the different spatial structures. The volcanic plume system, as described in section 5.1.1, is a stack of cylindrical slices and attached spheres. In comparison, the atmosphere system is a stack of horizontal 2D layered grids.

To transfer quantities from the plume to the atmosphere, we iterate over each atmosphere layer l. We find the plume slices that intersect the altitude range of l and select the slice s that is closest to the air layer's height h_l. For each cell i that is within $2r$ of the slice centre, we add quantities of moisture density M, water W, and temperature T. The addition is scaled by a linear function of the distance between the cell centre and the centre of the plume slice. When the distance is zero, the addition is equal to the difference between the value of s and the value of i, i.e., the cell takes the value of s. When the distance is greater than twice the radius of s, the addition is zero, i.e., the cell remains unchanged. This transfer depends only on the plume slices and does not consider the spheres or subspheres.

Compared with the less accurate transfer of moisture, temperature, and water, ash needs to closely correlate with the visual appearance of the plume. Since the plume's appearance is dictated by the spheres and subspheres, we therefore transfer ash from the spheres to the atmosphere. At each time step of the plume system, each sphere p calculates a projected density loss and adds it to an accumulated density loss A_p. The ash is transferred at the same time as the other quantities mentioned. For each sphere p, we find the closest grid cell i, and transfers A_p to i, resetting A_p to 0. Once in the atmospheric system, the ash is subject to precipitation as described in section 4.3.1.

To transfer quantities in the other direction, from the atmosphere to the plume, we find the closest slice s to each air layer l. We sample an annular region centred on s's position, which roughly corresponds to the area from which air is entrained into the eruptive column. In our implementation, this region has an inner radius of r_s, and an outer radius of $3r_s$. We calculate the mean values for velocity V, temperature T, and moisture density M across each cell in the annulus, with linear interpolation between the vertical layers above and below s. These values are then assigned to s.

Figure 6.1 provides an overview of all the transfers that happen between the different systems. Temperature T, moisture density M, and water density W are transferred in both directions between the plume and the atmosphere. The atmosphere transfers 2D horizontal velocity H to the plume, while the plume transfers ash density lost A to the atmosphere. The atmosphere transfers water density W, ash density A, and 3D velocity V to the offline rendering system, while the plume sends sphere ash A and 3D velocity V to the rendering system.

We transfer the quantities in order to mimic specific natural processes seen during a volcanic eruption. The temperature transfer in both directions is to reproduce the plume and atmosphere mixing that leads to localised atmospheric convection and the convective phase of the plume. We transfer moisture and water in both directions to model the entrainment and release of moist air leading to clouds. Horizontal 2D velocity is transferred from the atmosphere to the plume to ensure that the plume follows the downwind velocity field dynamically (compared to the static windfield used by Lastic et al.[Las+22] in their original implementation). Lastly, we transfer ash from the plume to the atmosphere to allow the ash to mix with water and model ash rain.

6.2 Procedural amplification

Our simulation models are sufficiently detailed to accurately portray the dynamics of the volcano plume and the atmosphere, as well as the interaction between the two subsystems. However, the appearance of clouds and plumes extends to a level of detail that the simulation cannot achieve in a reasonable amount of computation. We therefore incorporate details of finer scales by procedurally upsampling the data produced by the simulation.

We take advantage of the rich body of work in the field of procedural techniques[Lew84; Per85; PN01; Ola+02; CD05; BHN07; Lag+09; Lag+10] to support the upsampling with a wide array of auxiliary data, such as altitude, moisture, and velocity. In particular, by keeping the velocity field as part of the upsampling, it is possible to ensure temporal consistency in the upsampling between different frames by backtracking the velocity field through previous timesteps.

We perform the upsampling through carefully directed application of procedural noise to the data generated by the simulation. This is a field with a rich history and we cannot cover it in detail, instead referring the reader to a survey by Lagae et

al.[Lag+10]. For our purposes, we define procedural noise and give a short overview of the major methods, before describing our choice of procedural noise function and justifying the selection.

Noise as used in this context is random or pseudorandom values, usually with a probability density function that tends to be normal or Gaussian. *Procedural* noise functions refer to values that are generated by an algorithm or through code rather than from existing data sources such as an image. Some of the advantages of procedural noise functions, as discussed by Lagae et al.[Lag+10] and Ebert et al.[Ebe+02] make it especially suitable for our use.

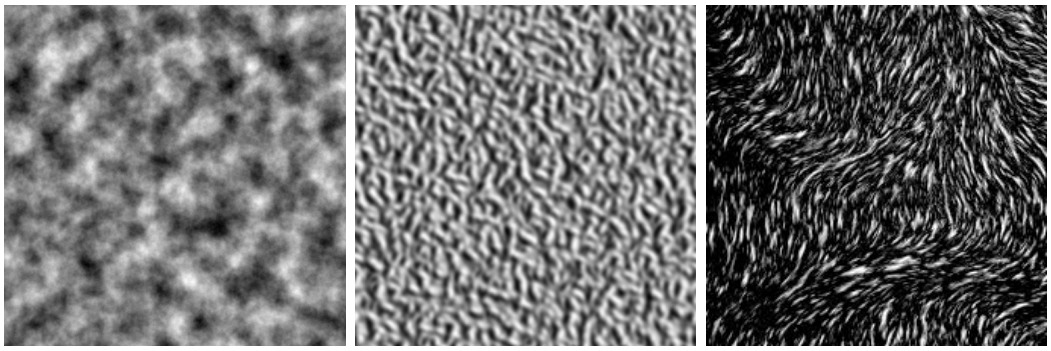
In particular, as *continuous* functions, we can use it at any scale without requiring base data at a specific resolution. By being *randomly accessible*, i.e., the function can be evaluated in constant time at any point on the lattice, it is suitable for use in parallel on multi-core CPUs or GPUs, leading to increased performance. Lastly, and perhaps most importantly, procedural noise functions are *parametrised*, allowing us to tune the noise for specific use cases without having to change the underlying function. This parametrisation also ensures that a particular result is reproducible given a specific seed value.

While the simplest noise generation algorithm is to produce a pseudorandom number sequence, the results are not realistic and are generally considered unsuitable for use in amplification in graphics. Instead, some form of coherent noise is preferred for its (usually) more realistic appearance. There are many methods for generating coherent noise and these are usually classified into three categories.

Lattice gradient noise functions generate noise through interpolating values or gradients at specific points on a lattice, usually the integer lattice. The defining example of this category is Perlin noise[Per85; Per02]. This was one of the first procedural noise generation functions and continues to be widely used in many aspects of graphics. It calculates a pseudorandom gradient for each point based on the hash of a lattice point, with values between lattice points being the spline interpolated result of the nearest gradients.

There are several variants of lattice gradient noise functions. Some are customised for specific physical phenomena, such as flow noise [PN01] or curl noise [BHN07] for advective motion and turbulent flow, respectively. Other variants are based on different lattices, such as simplex noise[Ola+02], which can lower computational complexity or eliminate directional artefacts.

Sparse convolution noise functions calculate noise as the sum of weighted kernels positioned randomly in the feature space. This was initially proposed by Lewis[Lew84; Lew89], who used a Poisson process convolved with a kernel to generate the noise. Other variants of sparse convolution noise include spot noise[Van91], Gabor noise[Lag+09], phasor noise[Tri+19], and texton noise[GLM17]. These use different kernel types for improved control and specific use cases, such as texture mapping on parametrised surfaces.



(1) Perlin[Per85] (2) Gabor[Lag+09] (3) Anisotropic[GZD08]

Figure 6.2: Examples of some 2D noise functions. From left to right, we show textures generated by Perlin noise[Per85], Gabor noise[Lag+09], and anisotropic noise[GZD08], covering the three major categories of noise functions.

Explicit noises are noise functions that are calculated beforehand and used as cached values at runtime. While not procedural techniques as defined by Lagae et al.[Lag+10], these are often used for reducing artefacts such as detail loss or interpolation bias. Wavelet noise[CD05] and anisotropic noise[GZD08] are examples of this category. It can also be used in conjunction with other methods, such as is the case with spot noise.

Figure 6.2 shows 2D examples of Perlin, Gabor, and anisotropic noises.

Given that the movement of the plume and clouds is largely advective in nature, we use *flow noise*[PN01] for the amplification. By varying the parameters of the function, we can easily tune the noise for the different types of clouds we consider. While the plume is turbulent as a whole, the scale of this turbulence is larger than the individual spheres and subspheres (see section 5.1.2) used by the simulation system. Therefore, the amplification does not need to portray inherent turbulence, as that is provided by the simulation data.

6.2.1 Cloud amplification

The cloud data is provided as the liquid water content W in each cell in the multiple, discrete 2D layers described in section 4.1. Any value W_i below a threshold $W_{\min}$ is considered to be below visibility, and therefore no clouds appear in that cell. We use a value of 0.1 $kg \cdot m^{-3}$ for $W_{\min}$. This value is derived from qualitative comparisons of different values, specific to the scenario in question. For example, if a lot of wispy, high clouds are desired, $W_{\min}$ can be reduced. If thick stratus clouds are the goal, $W_{\min}$ can be increased to make the difference in densities clearer.

To create a full volumetric density field, we linearly interpolate the liquid water content between the horizontal layers. A specific cloud type (cumulus, cirrus, or stratus) is then assigned to each cell based on the height of the layer and the convection in the cell[Vim+20]. We use a reduced subset of the cloudtypes used by Vimont et al., since the focus of the work is on the phenomena specific to volcanoes.

An issue arising from the linear upsampling of our layers is that the vertical gradients are weaker than the comparatively dense horizontal gradients. We can somewhat compensate for artefacts arising from this by laterally blurring the horizontal layers before the upsampling.

To add detail to our rendering, we warp the densities with flow noise[PN01], with parameters based on a combination of density values and the cloud type. To ensure temporal consistency between different steps, we can calculate a displacement $d(t)[x]$ back to the cloud source from a voxel position x. We recursively calculate $d(t)[x]$ using the 3D wind velocity $\vec{V}(t)$:

$$d(t + \Delta t)[x] = d(t)[x - \Delta t V(t)] - \Delta t V(t) \tag{6.1}$$

Over many timesteps, this backtracking can lead to discontinuities in the displacement and result in rendering artefacts. We can curb these discontinuities by relaxing the displacement, instead using a linear interpolation between $d(t + \Delta t)[x]$ and the identity. The interpolation is biased depending on the cloud type at the voxel x, as follows:

We add high frequency flow noise to **Cumulus** clouds to reflect the high turbulence present in these clouds. To sharpen the appearance of the cloud edges (see section 2.2.3 for an explanation), the density is scaled up before upsampling. To reflect the billowy appearance of cumulus clouds, the displacement interpolation is biased towards the identity.

Cirrus clouds are windswept and wispy. The displacement interpolation is biased towards $d(t + \Delta t)[x]$ to reflect this, while the flow noise is at a lower frequency than for cumulus clouds.

Stratus clouds are usually smooth sheets of clouds. The displacement interpolation is biased toward the identity, while the flow noise is applied at low frequency to reflect the low amount of turbulence in this type of cloud. To smooth out the appearance, the density is scaled down before upsampling.

6.2.2 Ash rain

Ash rain is usually characterised by strong striation in the vertical, corresponding to density spikes in the plume umbrella. Flow noise with a low frequency is added in the vertical (a period of 1000 to 3000 metres) and a high frequency in the horizontal (a period of 1 to 3 metres). This enables us to recreate the striated appearance. As with the water clouds, we backtrack a particular voxel (equation 6.1), though the velocity is scaled to be predominantly vertical. The horizontal velocity is matched to the prevailing 2D wind as calculated by the simulation.

6.2.3 Plume

In the simulation, the spheres and subspheres (section 5.1.2) are modelled as solid spheres of uniform density. This is difficult to represent in a volumetric rendering, and is inaccurate. The subspheres are also small and would require an impractically

dense voxel sampling to directly convert into a volumetric grid.

To overcome these challenges, we use a combination of a distance field, flow noise, and vector backtracking to create a voxel grid with voxels as cubes of roughly 100 metres on a side. For this, each voxel needs to store the distance d to the nearest plume sphere p, the ash density in the sphere A_p, and a displacement vector from the current position to the initial position of the sphere $\vec{D}_p$.

We use ray-tracing on the distance values of each voxel to approximate the hard transitions between the plume and the surrounding air. The opacity of this transition is then adjusted by the ash density of the sphere. To add detail, we subtract flow noise from the distance value of the voxel. The noise is generated with a period of 25/2 simulation steps to get the characteristic billowy appearance of the plume. This frequency is chosen to ensure that the frequency of the noise is less than the radius of the smallest sphere. As with the clouds, we need to maintain temporal consistency despite the random nature of the noise function. We do this by seeding the noise with the original position of the sphere derived from $\vec{D}_p$.

6.2.4 Lightning

The simulation data does not indicate when or where a lightning strike might occur. Instead, all lightning is created during the offline rendering step. At each rendering step, two spheres (or subspheres) in the plume with a high charge can connect to each other. We constrain the connections to occur probabilistically, with a probability of 0.1 at each time step, though this can be adjusted to reflect a more or less active plume. If the spheres connect, a lightning bolt is created and the charge in an area around the endpoints of the strike is neutralised. This neutralisation is a modification on top of the charge values provided by the simulation system (see section 5.3), meaning that multiple strikes can occur with the same sphere if the run continues long enough and the charge has time to accumulate again.

To render the final lightning strike, we use work from Reed et al.[RW94] as inspiration. We model lightning strikes with a branching wire structure with a single primary branch and multiple secondary branches radiating away from the primary. The secondary branches are randomly directed, though constrained to have an angle of less than 45 °compared to the primary branch.

To introduce further randomness and enhance the appearance of the lightning, we use noise to perturb the line segments that make up the wire structure. The noise function used is fractional Brownian motion noise[MV68], generated as a function of the spatial position and time for each line segment. This noise function is chosen because it generates a jagged appearance typical of lightning bolts.

Chapter 7

Results

In this chapter, we show the results produced by our system, assess the visual realism of our system, and discuss the results analytically.

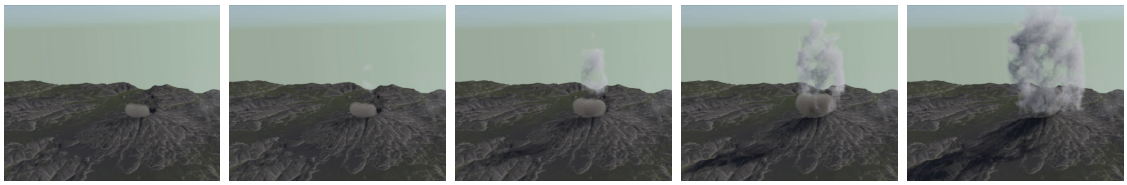
In terms of implementation, all of the simulation models are coded in C++20 [Com20], and compiled at maximum optimisation using the GNU compiler for C++ [Lea93], version 13.2. For previsualisation and tuning purposes we use OpenGL. The procedural upsampling and offline rendering is implemented in Houdini 19.5. Initial wind, moisture, and cloud fields were set for each layer using 2D Perlin noise [Per85] in the range $(-1, 1)$ added to a fixed value.

In our results, we find that our combined model can reproduce most of the visually striking phenomena arising from the interaction between the atmosphere and the volcanic plume. We also perform experiments to test the correspondence between the different submodels, and compare our system's behaviour to that of real volcanic eruptions.

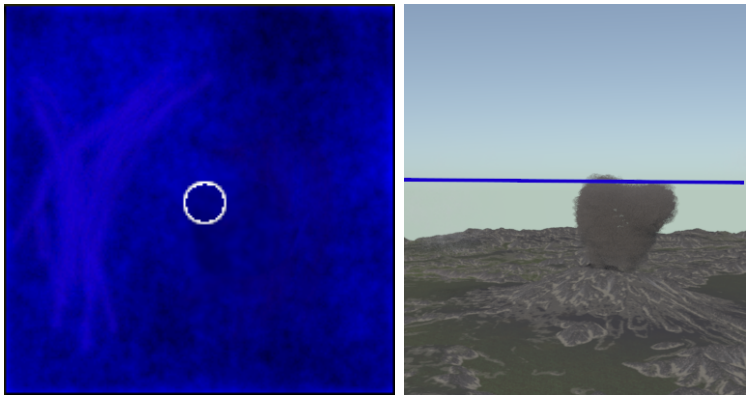
In terms of visual results, we capture the formation of transient condensation clouds, shown in figure 7.1. This shows a timelapse shortly after an explosion, ranging from $t = 0.3$s to $t = 6.8$s. The condensation cloud is localised to one side of the volcano vent due to the distribution of the ambient moisture levels in the atmosphere. This can be seen in figure 7.1(6), which shows a moisture map of the atmospheric layer at an altitude of 3000 metres, along with a visual indication of the layer's position relative to the volcanic plume.

Our system successfully generates embedded clouds in the plume. Figure 7.2 shows three skirt layers being driven up and then deformed by the volcanic plume. This happens over a time span of $t = 60$s. Cap clouds, which are a result of the same processes at different heights and scales, are shown in figure 7.3 over a time span of $t = 120s$. A single pileus cloud is driven ahead of the plume before reaching neutral buoyancy, where the plume then breaks through the cap cloud.

We also provide results that demonstrate ash rain in an eruption that extends to the umbrella phase. It must be noted that while ash fall happens during every stage of the eruption, it is usually obscured by the plume in the earlier stages. Figure 7.4 shows ash rain precipitating from the plume umbrella, over a time span of $t = 510$s,

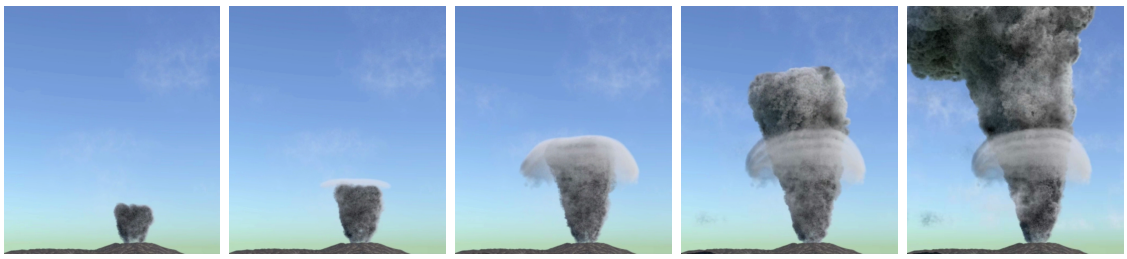


(1) $t = 0.3$s (2) $t = 1.3$s (3) $t = 2.3$s (4) $t = 4.3$s (5) $t = 6.8$s



(6) $t = 0.6s$ (7) $t = 0.6s$

Figure 7.1: [top] A timelapse showing the formation of a transient condensation cloud. The condensation cloud is localised to one side (towards the camera) due to more ambient moisture being present in that area of the atmosphere. [bottom] On the left, a moisture map at $t = 0.6s$ with a narrow field of view around the volcano (indicated by the white circle), at an altitude of 3000 metres. Lighter blue indicates higher moisture content, particularly visible left of the vent (towards the camera in the top renders). On the right, the blue line indicates the location of the moisture layer in (6) relative to the plume.



(1) $t = 20$s (2) $t = 30$s (3) $t = 40$s (4) $t = 50$s (5) $t = 80$s

Figure 7.2: Three skirt cloud layers deformed and driven ahead of the plume, condensing due to adabiatic cooling.

(1) $t = 120$s (2) $t = 140$s (3) $t = 200$s (4) $t = 240$s (5) $t = 264$s

Figure 7.3: A single cap cloud forms at the head of the plume, rising with it until reaching a neutral buoyancy height. The plume then pushes through the cap cloud.

(1) $t = 450$s (2) $t = 540$s (3) $t = 630$s (4) $t = 840$s (5) $t = 960$s

Figure 7.4: Fine ash mixes with water in the umbrella phase, before precipitating out as ash rain.

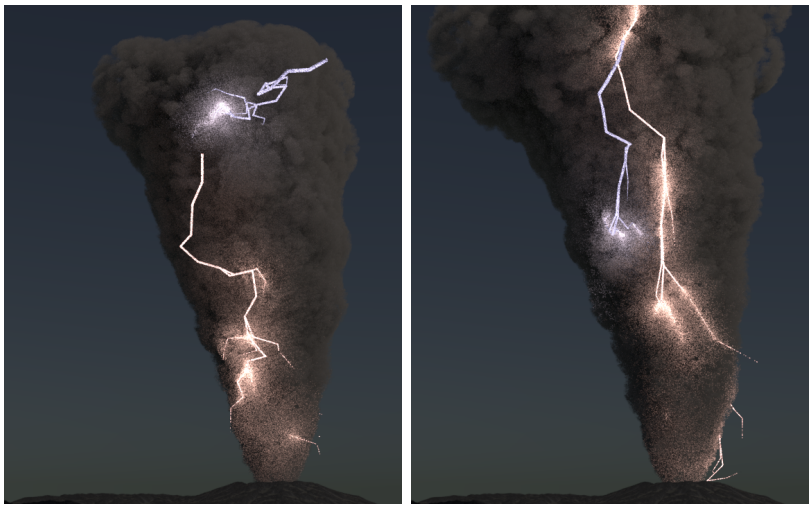


Figure 7.5: Volcanic lightning strikes.

starting from $t = 450$s. The initial wind field is set to $10 \pm 1\ m \cdot s^{-1}$ east to west (left to right in the render) above an altitude of $7000m$, and $10 \pm 1\ m \cdot s^{-1}$ south to north (from front to back in the render) below that altitude.

Lastly, we show volcanic lightning in the plume in figure 7.5. Our system produces lightning at a peak frequency of 18 visible strikes per minute (measured across 10 frames, 30 seconds of simulation time) at time $t = 450$s. This compares favourably with the 27 strikes per minute from real eruptions[Beh+13], which was measured using low frequency radio waves generated by the lightning strikes and thus includes the strikes obscured by the plume itself. This also validates our choice of 0.1 for the connection probability per time step (see section 6.2.4).

7.1 Validation

To confirm that our models conform to real world data, we perform a number of experiments. In particular, we consider ash transmission through the atmosphere, ash deposition onto the terrain, the progressive electrical charge build up in the plume spheres, and an ablation study comparing the advected wind field with a uniform wind field.

7.1.1 Ash rain

We assess the accuracy of the ash rain with two methods. First, we investigate the behaviour of the ash during transfer from the plume model and in the atmospheric layers after the transfer. As part of this, we keep track of the quantity of ash deposited onto the terrain. As shown in figure 7.6(1), substantial ash accumulates around the vent from early in the eruption, even before the umbrella forms. This is to be expected since the plume undergoes a continuous density loss, regardless of the phase of the eruption. There is little or no ash accumulation on the vent itself, kept aloft by the intense uplift in the eruptive column.

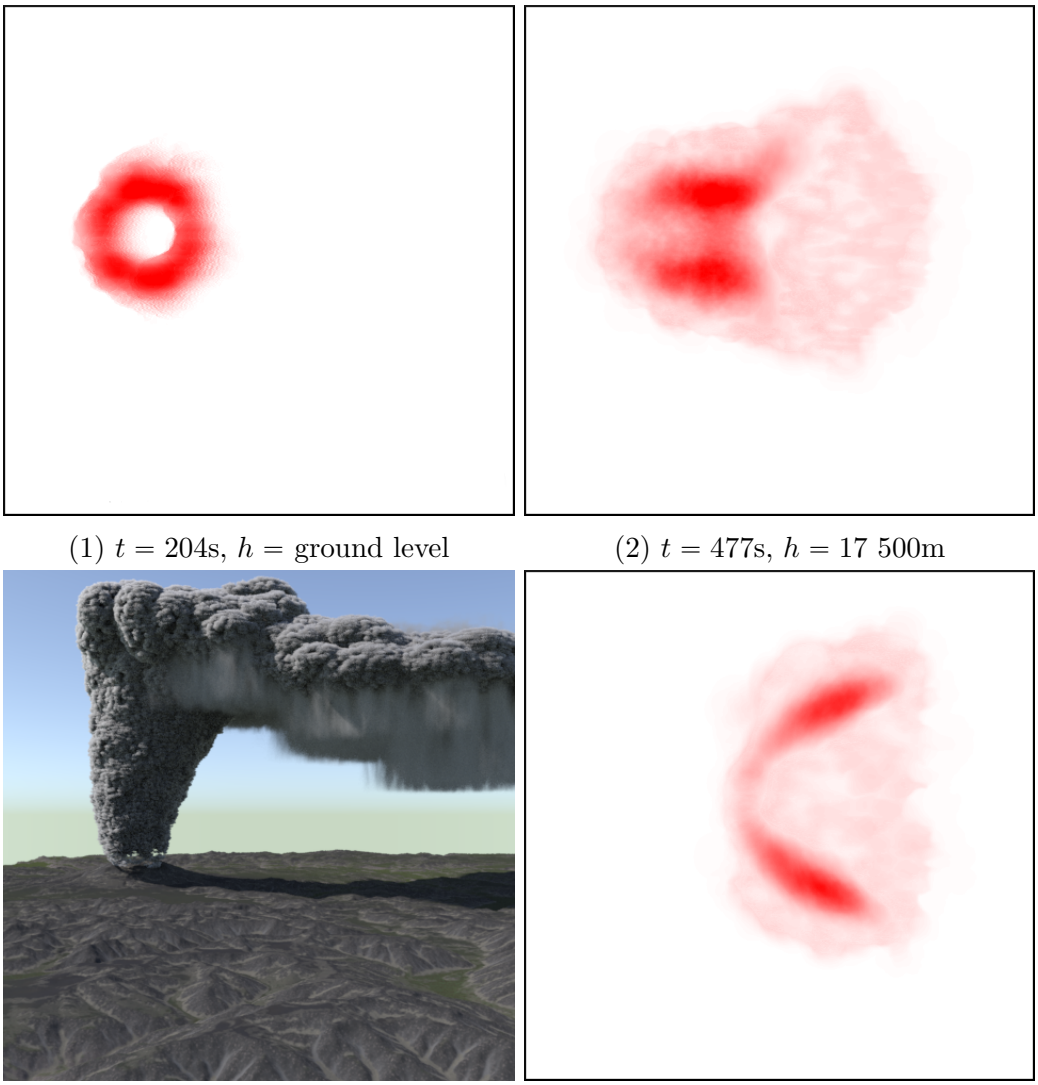
The scenario that figure 7.6 is extracted from has an initial windfield that is consistent from west to east (i.e., left to right in the images). Figure 7.6(2) shows ash density in the topmost atmospheric layer below the umbrella, at a height of 17.5 km. There is a clear visual correlation between the plume in figure 7.6(3) and the distribution of ash in 7.6(2). As ash accumulates in one layer, it will fall to the layer below if sufficiently dense (see section 4.3.1). Figure 7.6(4) shows the ash density in the air layer immediately below 7.6(2), with an altitude of 16.2 km, at the same time step. Given the consistent westerly wind, the ash that has been taken up by the atmospheric system is more dispersed than the plume and further away from the plume.

The second evaluation metric is the longer term deposition of ash on the terrain. This is a fairly common metric in vulcanology, and we can compare the results of our simulation with real-world data. In figure 7.7, we show the accumulated ash depth along transects shown on the map on the right after the first 20 minutes of the eruption, with a fairly uniform wind field from the north west (i.e., the top left corner). The deposition profile is, as expected, heavily influenced by the wind strength and direction.

While eruptions in the reality can result in ash depths of multiple metres, this is the result of eruptions that continue for many days or even weeks. We find the peak ash depth to be some distance away from the vent in the general wind direction, which is a consequence of modelling only the fine ash that tend to get carried along by the convective column. Coarser grains are denser and tend to precipitate out early in the plume life cycle. Despite this, our model exhibits an exponential thinning pattern that agrees well with field studies and vulcanological models[Bra+83; Pyl89; BES98], as shown in figure 7.8. Note the logarithmic scale on the y-axis. The x-axis uses the square root of isopach area, that is, the area enclosed by a contour line of equal depth. The linear relation of $ln(T)$ to $\sqrt{A}$, shown in the graphs, corresponds to an exponential relation of T to D[Pyl89]. In this case, T refers to the thickness of the ash deposition, A the isopach area, and D the distance from the vent.

7.1.2 Lightning

To evaluate the accuracy of the lightning, we visualise the charge in the spheres of the plume. Figure 7.9 shows the incremental and accumulated charge for a plume that is transitioning into the umbrella stage. We show that charge accumulation greatly increases above the -20 °C isotherm, indicated by the black outline at an altitude of



(1) $t = 204$s, $h =$ ground level (2) $t = 477$s, $h = 17\ 500$m

(3) $t = 477$s (4) $t = 477$s, $h = 16\ 200$m

Figure 7.6: Ash density in the atmosphere with a wind from left to right. Darker red indicates a higher density. (1) Ash accumulated on the terrain prior to the plume entering the umbrella phase. (2) Ash density in the top most layer below the plume umbrella, where the majority of ash transfer from the plume to the atmosphere takes place. (3) Scene render of the plume and umbrella, showing ash rain precipitating from the umbrella. (4) Ash density in the layer immediately below (2), showing dispersal away from the plume according to the wind field.

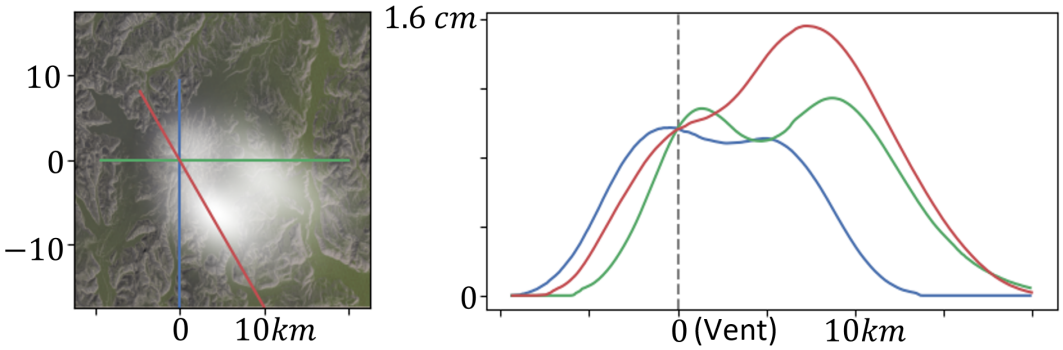


Figure 7.7: Ash deposition on the terrain 20 minutes after eruption. [left] Ash accumulation (white is deeper ash) and colour-coded transects overlaid on the landscape. [right] Ash depth along the transects graphed as a function of distance from the volcano vent.

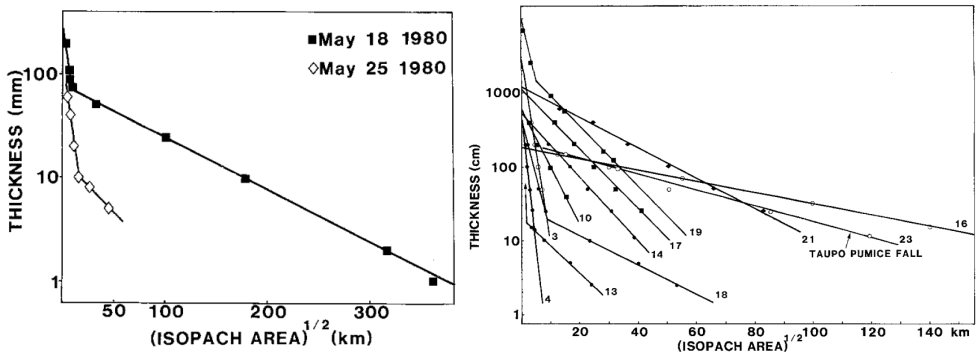


Figure 7.8: Graphs of ash deposition from real volcanic eruptions, sourced from Pyle[Pyl89]. Note the logarithmic scale on the y-axis. The x-axis uses the square root of isopach area, that is, the area enclosed by a contour line of equal depth. This is proportional to the distance from the peak depth.

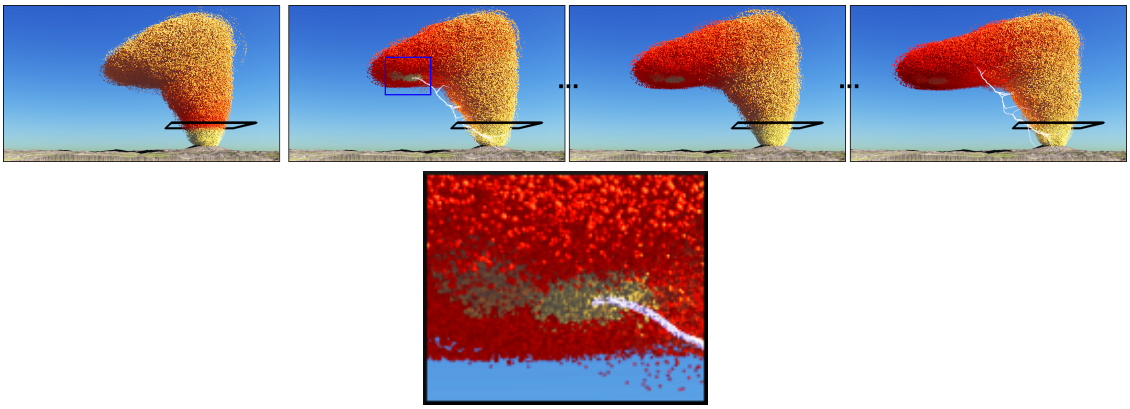


Figure 7.9: The electrical charge in each sphere visualised, with darker red indicating a greater charge and light yellow indicating no charge. The black frame is at an altitude of 4100 m above sea level, and roughly corresponds to the -20 °C isotherm. [top left] The increase in charge over a single time step. Note the significant increase above the -20 °C isotherm. [top right] Total accumulated charge in each sphere. [bottom] Expanded view of the region in the blue square. Note the regions of low charge (lighter) that result from discharge as a result of lightning strikes.

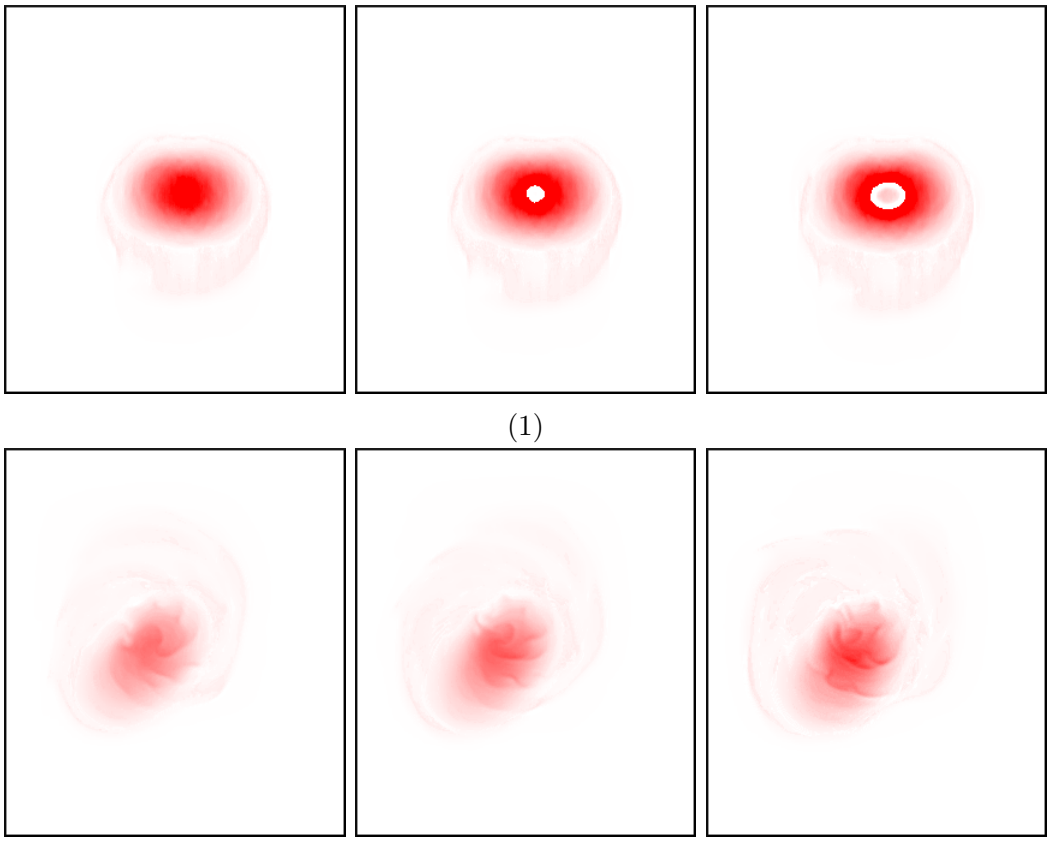
4100 m. This is to be expected given the two differing regimes of charge accumulation (see equation 5.2). We also show how the accumulated charge increases over time for all spheres. The spheres in the vicinity of a lightning strike are discharged, and this can be seen as areas of low charge in the visualisation.

7.1.3 Windfield ablation

The atmospheric sub-model, particularly the advected windfield, requires significant computation and adds considerable design complexity. Thus, it is worth evaluating if the benefits outweighs this performance and implementation cost. We perform an ablation study comparing the impact of an advected windfield against a uniform wind field. Figure 7.10 depicts the ash density in the atmosphere under a constant, uniform wind field, and under a complex, advected wind field. Both cases utilise a grid resolution of 512 by 512 cells per air layer. 7.10(1) shows a constant wind field at 10 $m \cdot s^{-1}$ from north to south (top to bottom), with the ash dispersing slowly over time. In contrast, 7.10(2) shows an advected wind field, with eddies and turbulence being reflected in the ash density. This level of detail is not possible with a uniform wind field.

7.2 Performance

We perform timing tests on an Intel® Core™ i9-9900K CPU @ 5GHz and an NVidia® RTX 2080M (8GB). Table 7.1 shows the computational performance for each scenario in our simulation. Table 7.5 depicts the computational performance for each sub-model in the same scenarios. We found that, for a single simulation step, the atmospheric model takes much longer than a step for either the plume or embedded cloud models. However, the atmospheric model is run at a significantly lower frequency, which partly mitigates this issue.



(1)

(2)

Figure 7.10: An ablation study comparing the ash distribution over a time span of $t = 180$s, under a uniform wind field of 10 $m \cdot s^{-1}$ north to south (top to bottom) [top]; and under an advected wind field [bottom]. The advected wind field rotates in a clockwise fashion (as set by the initial wind field), and displays eddies and turbulence that are impossible under a uniform wind field.

Scenario	Scene Dimensions	Grid size	Layers	Time step computation (in s)
Skirt (figure 7.2)	60 × 60 × 20km	512 × 512	20	107.2s
Cap & condensation (figures 7.1 & 7.3)	60 × 60 × 20km	512 × 512	20	59.6s
Ash rain (figures 7.4 and)	60 × 60 × 20km	1024 × 1024	20	79.4s

Table 7.1: System performance for 3 scenarios. The average performance for a simulation time of $t = 1$s is reported, with the average calculated across 320 steps. Each scenario used the same layer structure: layers at 0.4, 1.2, 2.0, 2.7, 3.0, 3.5, 4.1, 4.7, 5.5, and 6.5 km, with the remainder evenly spaced up to 20 km.

Resolution	**512 ×512**		**1024 ×1024**	
Component	Percentage	Compute time	Percentage	Compute time
Advection	47.45	4.038s	25.87	13.41s
Pressure solve	39.33	3.347s	62.56	32.42s
Terrain heating	0.000002	0.0000002s	0.0000004	0.0000021s
Atmospheric heating	0.42	0.036s	0.26	0.1348s
Moisture uptake	0.57	0.049s	0.30	0.1555s
Obstacle tamping	0.0007	0.00006s	0.0003	0.0016s
Phase transition	4.7	0.4s	2.51	1.301s
Precipitation	1.01	0.086s	1.72	0.8915s
Vertical coupling	6.51	0.554s	6.78	3.514s
Total:	100%	8.51s	100%	51.83s

Table 7.2: The computation time spent on each component of the atmosphere sub-model per time step, with 20 layers on a grid of 512 × 512 cells. For larger grids, the pressure solve takes up a larger proportion, up to 70% on a 1024 × 1024 grid.

Within the atmosphere sub-model, we found that the computation is dominated by the advection and pressure solve, which requires 90% of this sub-model's compute time. At higher resolutions, the pressure solve alone requires proportionally more time, up to 63% (from 40%) of total compute for the atmospheric submodel on a 1024 × 1024 grid. Table 7.2 shows the average computational performance and proportion of the total for each component, using the ash rain scenario on a 512×512 and a 1024 × 1024 grid. Adding more layers and increasing the grid resolution linearly increases computation time, as shown in table 7.3. While larger grid sizes (2048 × 2048) were tested, the computation times required proved infeasible and had limited visual effect on the final results. In practice, we found that around 20 layers with a cell size between 60 and 100 metres on a side is enough to capture the relevant atmospheric behaviour around the volcanic plume.

		Resolution		
		256^2	512^2	1024^2
Layers	10	0.86s	3.65s	21.64s
	20	1.66s	8.51s	51.83s
	30	3.12s	15.23s	98.5s

Table 7.3: Total performance requirement for each time step (Δt = 1s) of the atmospheric sub-model as a function of the number of layers and layer resolution, using the ash rain scenario.

Despite the grid sizes being equal for both the skirt and cap and condensation cloud scenarios, the atmosphere sub-model requires more than double the computation time for the cap and condensation cloud scenario. This is a consequence of running the phase transition component at a much higher frequency (Δt = 0.1s) compared to the rest of the sub-model, which is run with a time step of Δt = 2s. In the other scenarios, all components of the atmosphere system run at the same frequency, with Δt = 2s and Δt = 3s for the skirt and ash rain scenarios, respectively.

Table 7.4 shows the total time required to compute the scenarios for 320 steps. This is sufficient to produce all the effects demonstrated in this chapter, though the simulations can vary in length if only some effects are desired (e.g., less simulation time is required for the skirts).

Scenario	Average computation time for 1 second of simulation time	Δt for atmosphere system	Computation time for full scenario
Skirt	107.2s	2s	1143.46 minutes
Cap & condensation	59.6s	2s	635.73 minutes
Ash rain	79.4s	3s	1270.4 minutes

Table 7.4: The total computation time required for each of the three scenarios listed in table 7.1, for 320 simulation steps. The atmospheric sub-model uses different timesteps depending on the scenario, while the plume sub-model always uses a timestep of $\Delta t = 0.02$s.

	Timestep computation (in s)			
Scenario	Plume	Embedded Clouds	Atmosphere	Total
Skirt	22s	76.5s	8.7s	107.2s
Cap & condensation	5.5s	35s	19.1s	59.6s
Ash rain	26.5s	Not applicable	52.9s	79.4s

Table 7.5: Average performance for each component in the three scenarios listed in table 7.1, for a simulation time of $t = 1$s. The ash rain scenario had no embedded clouds in it, hence no cost associated with that sub-model.

The computation requirements for the plume sub-model increases in a linear fashion as the number of active slices and spheres grow as part of the eruption. When the number of slices and spheres is low, the cost for this sub-model can be negligible. However, when the plume is well developed (e.g., $t = 600$s), this model can require up to 1.2 seconds of compute for each time step ($\Delta t = 0.02$s). The timings in table 7.5 refer to the computation required for 50 steps of the plume system, for a simulation time of $t = 1$s, averaged over a complete run of the scenario (between 20 and 40 minutes of simulated time). Figure 7.11 shows the computation time required per time step as a function of the simulation time, showing a linear increase for the plume subsystem while the atmospheric subsystem remains nearly constant. This increase over time is especially noticeable in the average time required for plume sub-model in the ash rain scenario, which contributes to the large computation time required for the full ash rain scenario.

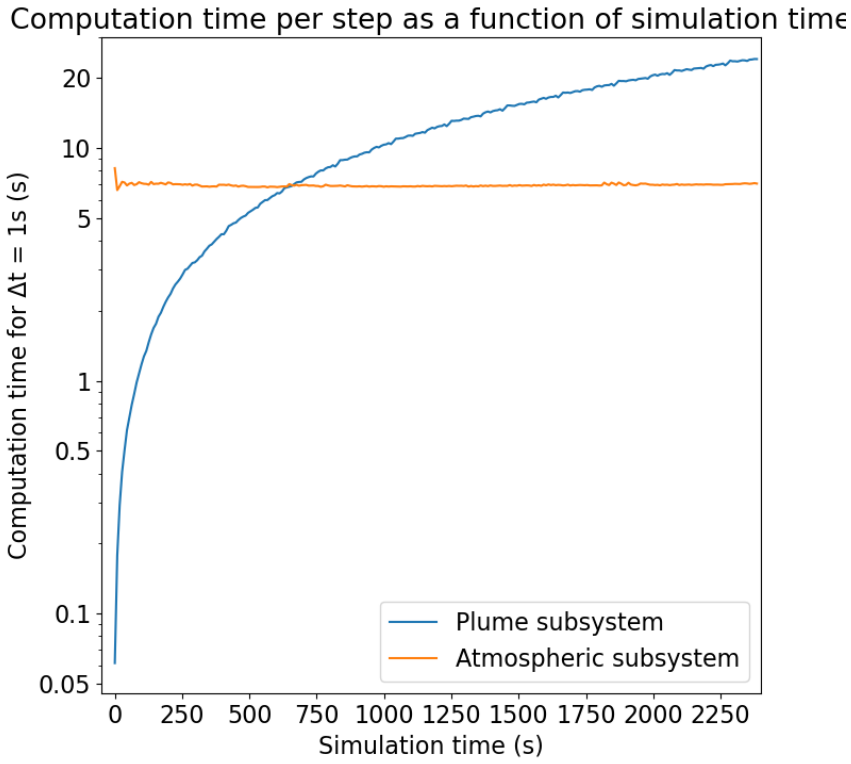


Figure 7.11: We graph the computation time required per time step as a function of simulation time. This data is extracted from the ash rain scenario (figure 7.4), on a grid of 512×512, 20 air layers, and $\Delta t = 0.02$s for the plume subsystem and $\Delta t = 3$s for the atmospheric subsystem. Note the logarithmic scale on the y-axis. The plume subsystem increases linearly in computation cost as the eruption progresses, while the atmospheric subsystem remains fairly constant in its computation cost. The atmospheric system requires a few steps before the conjugate gradient solver stabilises, which explains the extra computation initially.

The skirt sub-model runs in lockstep with the plume sub-model, with a time step of $\Delta t = 0.02$s. Calculating the deformation of the skirt surfaces is particularly expensive since it requires finely subsampling the plume trajectory. In future, a more sophisticated scheme for calculating the intersection of the skirt sheet and the plume could reduce the computation required. The computation cost is also mitigated by dropping intersection testing once the skirt sheet has reached neutral buoyancy.

7.3 Limitations

Our solution for coupling explosive volcanic eruptions with the atmosphere presents a reasonable compromise between performance and accuracy. However, it remains far slower than real time. The atmospheric sub-model remains fairly coarse, with cells ranging in size from 60 metres to 120 metres on a side, which can lead to some inaccuracies near the plume. While finer resolutions are possible, this incurs a significant performance cost and is infeasible for simulation times of $t = 60$s or

more. Given that explosive volcanic eruptions range in length from tens of minutes to hours or even days, our solution is limited to a coarse resolution. A GPU implementation or a multiresolution grid for the atmospheric sub-model could both serve as a potential mitigation of this issue. The algorithms in use are suitable for both, and it is likely to significantly reduce the computation time required.

A consequence of the largely procedural charge generation (since we do not explicitly model friction) is that we do not differentiate between positive and negative charge. Highly charged areas are connected for each lightning strike, which, while visually acceptable, is not physically accurate.

Similarly, the coupling between the horizontal layers in the atmosphere is procedural. This requires forgoing physical accuracy in a number of places elsewhere as well. The pressure values used in any of the phase transition equations (see section 4.2.6) uses a very coarse, barometric approximation, rather than the true value. Similarly, the pressure values used around the transient condensation cloud are barometric, requiring a workaround to mimic the visual appearance. The physical accuracy of the layer exchange could be improved by using a more accurate, full 3D model of the atmosphere, though this would come at significant computational cost.

Lastly, as is the case with many simulation methods that involve fluid dynamics, setting up the initial conditions and tuning the parameters requires time. This procedure usually requires some experimentation and adaptation. In our case, specifically, the number of air layers and their altitudes need to be carefully set to match atmospheric features such as inversion layers. Additionally, the vertical sampling near the volcano vent must be dense enough for the plume to be stable, with exact placement requiring experimentation.

Chapter 8

Conclusion

Volcanic eruptions are visually striking phenomena and play an important role in digital media such as films or games. However, existing solutions for animating volcanic plumes are usually bespoke, one-off solutions, while existing plume models ([MDN02b; Miz+03; Las+22]) do not consider dynamic atmospheres, limiting their physical accuracy and the phenomena that can be depicted. A naive combination with existing cloud models would also not suffice considering the extremes of both temperature and altitude that are present in volcanic plumes.

To address this, we conceived and have developed an integrated framework to generate and animate explosive volcanic eruptions. We performed a detailed investigation of the physical processes that determine the appearance of skyscapes around volcanic plumes, with a focus on the movement in the atmosphere, cloud formation, and the formation of ash rain. We also did an analysis of the dynamics of incompressible fluids, required for implementing our treatment of fluid dynamics.

We compiled a thorough exploration of the state of the art in this field, with a focus on cloud modelling techniques, and the rise of hybrid models, i.e., models combining more than one modelling technique. We also conducted a survey of volcanic plume models in both computer graphics and geophysics, finding that while little attention has been given to these phenomena in computer graphics, it is a very active field of research in the geophysical sciences.

We adapted Vimont et al.'s[Vim+20] cloud model for use in our system. We made changes in order to accommodate the temperature and height extremes encountered around a volcanic plume, and implement our own treatment of fluid dynamics to address the inaccuracies around solid boundaries that Vimont et al.'s system produced. We also modified Lastic et al.'s[Las+22] plume model to accurately portray ash rain and volcanic lightning, while implementing our own model of lightning and embedded clouds in the plume.

We combined the two models through bi-directional interaction, transferring physical quantities such as ash, moisture, and temperature between the two submodels. By running the submodels at different frequencies, we addressed some of the pitfalls that could arise from the high energy dynamics encountered in and around the plume. Through the use of procedural amplification and procedural

noise functions, we enhance the final renders significantly.

We demonstrated the graphical depiction of several previously unsupported phenomena. This includes ash rain, embedded clouds, transient condensation clouds, and volcanic lightning. Moreover, the dynamic atmosphere allows the plume to track crosswinds and turbulent eddies more realistically. We validate our contributions by visually comparing our results to that of real volcanic eruptions, as well as assessing the ash deposition patterns of our model compared to that of real eruptions. We also justified our design decisions through an ablation study, demonstrating the necessity of our combined system.

Despite the high energy dynamics present, the fluid solver for the atmospheric simulation maintains indefinite stability. Computation times depend on the number of layers and the resolution of said layers, as well as the number of spheres in the plume. Generally, less than 2 minutes of computation is required for each second of simulation. Since many of the computationally intensive components (such as advection) are suitable for parallel execution[CLT07], a performance boost would be possible through GPU implementation. All of the weather components, except for the advection and the vertical inflow outflow, are independent on a per cell basis, allowing for SIMD vectorisation, which is especially suited to a GPU implementation. By using a memory buffer, the advection component can be vectorised as well.

This work also opens up opportunities for future extensions in volcano simulation for computer graphics. Combining our model with models for terrain deformation (e.g., [CFM11] or [Pey+09]) and lava flows (e.g., [Zha+17]) would allow for a single framework simulating all aspects of a volcanic eruption. A particularly challenging facet would be to model lahar flows, a mixture of pyroclasts, water, and debris that can vary significantly in density and viscosity across its volume. Lava flows, which have been modelled in computer graphics[Sto+99; Zha+17], are fairly homogeneous in comparison and smoothly vary in temperature and density. A possible approach would be a non-homogenous Lagrangian approach, with the different components of lahars each having separate particle parameters.

The noise functions used in this work are largely retained from the previous systems, and there are other options available as well, such as Voronoi cell noise[Wor96]. A more detailed analysis and comparison of potential noise functions could be employed to determine the noise functions best suited for this specific purpose.

Another possibility would be to incorporate ash rain and aerosolised ash in the atmosphere into weather systems, with the accompanying broad-scale effects. This would be of particular interest in games with seasonal cycles, such as *Medieval Dynasty* or *Red Dead Redemption 2*, or lightweight global climate simulations suitable for use in education.

Bibliography

[Ada+07] Bart Adams, Mark Pauly, Richard Keiser, and Leonidas J Guibas. "Adaptively sampled particle fluids". In: *ACM SIGGRAPH 2007 papers*. 2007, 48–es.

[AJ05] Cristina L Archer and Mark Z Jacobson. "Evaluation of global wind power". In: *Journal of Geophysical Research: Atmospheres* 110.D12 (2005).

[Atm76] US Standard Atmosphere. *US standard atmosphere*. National Oceanic and Atmospheric Administration, 1976.

[Bar+68] JC Barnes, AH Glasen, PE Sherr, and JH Willand. "World-wide cloud cover distribution for use in computer simulations Final report, 18 Jan. 1967-17 Jan. 1968". In: (1968).

[Bar13] Roger G Barry. *Mountain weather and climate*. Routledge, 2013.

[Bar82] Sumner Barr. "Skirt clouds associated with the Soufriere eruption of 17 April 1979". In: *Science* 216.4550 (1982), pp. 1111–1112.

[BBD12] RJ Brown, C Bonadonna, and AJ Durant. "A review of volcanic ash aggregation". In: *Physics and Chemistry of the Earth, Parts a/b/c* 45 (2012), pp. 65–78.

[BC09] Roger G Barry and Richard J Chorley. *Atmosphere, weather and climate*. Routledge, 2009.

[Beh+13] Sonja A Behnke, Ronald J Thomas, Stephen R McNutt, David J Schneider, Paul R Krehbiel, William Rison, and Harald E Edens. "Observations of volcanic lightning during the 2009 eruption of Redoubt Volcano". In: *Journal of Volcanology and Geothermal Research* 259 (2013), pp. 214–234.

[BES98] Costanza Bonadonna, GGJ Ernst, and RSJ Sparks. "Thickness variations and volume estimates of tephra fall deposits: the importance of particle Reynolds number". In: *Journal of Volcanology and Geothermal Research* 81.3-4 (1998), pp. 173–187.

[BHN07] Robert Bridson, Jim Houriham, and Marcus Nordenstam. "Curl-noise for procedural fluid flow". In: *ACM Transactions on Graphics (ToG)* 26.3 (2007), 46–es.

[BHR07] Henry E Bass, Claus H Hetzer, and Richard Raspet. "On the speed of sound in the atmosphere as a function of altitude and frequency". In: *Journal of Geophysical Research: Atmospheres* 112.D15 (2007).

[Bi+16] Shuoben Bi, Shengjie Bi, Xiaowen Zeng, Yuan Lu, and Hao Zhou. "3-Dimensional modeling and simulation of the cloud based on cellular automata and particle system". In: *ISPRS International Journal of Geo-Information* 5.6 (2016), p. 86.

[Bon+15] Costanza Bonadonna, Marco Pistolesi, Raffaello Cioni, Wim Degruyter, Manuela Elissondo, and Valerie Baumann. "Dynamics of wind-affected volcanic plumes: The example of the 2011 Cordón Caulle eruption, Chile". In: *Journal of Geophysical Research: Solid Earth* 120.4 (2015), pp. 2242–2261.

[Bra+83] S Brazier, RSJ Sparks, SN Carey, H Sigurdsson, and JA Westgate. "Bimodal grain size distribution and secondary thickening in air-fall ash layers". In: *Nature* 301.5896 (1983), pp. 115–119.

[Bri15] Robert Bridson. *Fluid simulation for computer graphics*. CRC press, 2015.

[Buc96] AL Buck. "Buck Research CR-1A User's Manual". In: *Buck Research Instruments: Boulder, CO, USA* (1996).

[CD05] Robert L Cook and Tony DeRose. "Wavelet noise". In: *ACM Transactions on Graphics (TOG)* 24.3 (2005), pp. 803–811.

[CEB16] Matteo Cerminara, Tomaso Esposti Ongaro, and Luigi C Berselli. "ASHEE-1.0: a compressible, equilibrium–Eulerian model for volcanic ash plumes". In: *Geoscientific Model Development* 9.2 (2016), pp. 697–730.

[CFM11] Justin Crause, Andrew Flower, and Patrick Marais. "A system for real-time deformable terrain". In: *Proceedings of the South African Institute of Computer Scientists and Information Technologists Conference on Knowledge, Innovation and Leadership in a Diverse, Multidisciplinary Environment*. 2011, pp. 77–86.

[CG22] Corrado Cimarelli and Kimberly Genareau. "A review of volcanic electrification of the atmosphere and volcanic lightning". In: *Journal of Volcanology and Geothermal Research* 422 (2022), p. 107449.

[Che+21] Qian Chen, Yue Wang, Hui Wang, and Xubo Yang. "Data-driven simulation in fluids animation: A survey". In: *Virtual Reality & Intelligent Hardware* 3.2 (2021), pp. 87–104.

[Cim+14] Corrado Cimarelli, MA Alatorre-Ibargüengoitia, Ulrich Kueppers, Bettina Scheu, and Donald B Dingwell. "Experimental generation of volcanic lightning". In: *Geology* 42.1 (2014), pp. 79–82.

[CLT07] Keenan Crane, Ignacio Llamas, and Sarah Tariq. "Real-time simulation and rendering of 3d fluids". In: *GPU gems* 3.1 (2007), pp. 633–675.

[Com20] C++ Standards Committee. *Programming languages – C++*. ISO/IEC 14882:2020. Geneva, Switzerland: International Organization for Standardization and International Electrotechnical Commission, Dec. 2020.

[CON16] Matteo Cerminara, Tomaso Esposti Ongaro, and Augusto Neri. "Large Eddy Simulation of gas–particle kinematic decoupling and turbulent entrainment in volcanic plumes". In: *Journal of Volcanology and Geothermal Research* 326 (2016), pp. 143–171.

[Cou+12] J Courtney, Steve Buchan, Randall S Cerveny, Pierre Bessemoulin, Thomas C Peterson, JM Rubiera Torres, John Beven, John King, Blair Trewin, and Kenneth Rancourt. "Documentation and verification of the world extreme wind gust record: 113.3 m s-1 on Barrow Island, Australia, during passage of tropical cyclone Olivia". In: *Australian Meteorological and Oceanographic Journal* 62.1 (2012), p. 1.

[CT17] Mengyu Chu and Nils Thuerey. "Data-driven synthesis of smoke flows with CNN-based feature descriptors". In: *ACM Transactions on Graphics (TOG)* 36.4 (2017), pp. 1–14.

[CW12] Ray Cas and James Wright. *Volcanic successions modern and ancient: A geological approach to processes, products and successions*. Springer Science & Business Media, 2012.

[Den+23] Yitong Deng, Hong-Xing Yu, Diyang Zhang, Jiajun Wu, and Bo Zhu. "Fluid Simulation on Neural Flow Maps". In: *ACM Transactions on Graphics (TOG)* 42.6 (2023), pp. 1–21.

[DG17] Rui PM Duarte and Abel JP Gomes. "Real-time simulation of cumulus clouds through SkewT/LogP diagrams". In: *Computers & Graphics* 67 (2017), pp. 103–114.

[DNM93] Flavio Dobran, Augusto Neri, and Giovanni Macedonio. "Numerical simulation of collapsing volcanic columns". In: *Journal of Geophysical Research: Solid Earth* 98.B3 (1993), pp. 4231–4259.

[Dob+00] Yoshinori Dobashi, Kazufumi Kaneda, Hideo Yamashita, Tsuyoshi Okita, and Tomoyuki Nishita. "A simple, efficient method for realistic animation of clouds". In: *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. 2000, pp. 19–28.

[Dob+99] Yoshinori Dobashi, Tomoyuki Nishita, Hideo Yamashita, and Tsuyoshi Okita. "Using metaballs to modeling and animate clouds from satellite images". In: *The Visual Computer* 15.9 (1999), pp. 471–482.

[DR06] F Dobran and JI Ramos. "Global volcanic simulation: physical modeling, numerics, and computer implementation". In: *Developments in Volcanology*. Vol. 8. Elsevier, 2006, pp. 311–372.

[Dry+22] Claudia Drygala, Benjamin Winhart, Francesca di Mare, and Hanno Gottschalk. "Generative modeling of turbulence". In: *Physics of Fluids* 34.3 (2022).

[DSY10] Yoshinori Dobashi, Yusuke Shinzo, and Tsuyoshi Yamamoto. "Modeling of clouds from a single photograph". In: *Computer Graphics Forum*. Vol. 29. 7. Wiley Online Library. 2010, pp. 2083–2090.

[DYN02] Yoshinori Dobashi, Tsuyoshi Yamamoto, and Tomoyuki Nishita. "Interactive rendering of atmospheric scattering effects using graphics hardware". In: *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*. 2002, pp. 99–107.

[DYN06] Yoshinori Dobashi, Tsuyoshi Yamamoto, and Tomoyuki Nishita. "A controllable method for animation of earth-scale clouds". In: *Proc. of CASA* (2006), pp. 43–52.

[DYN10] Yoshinori Dobashi, Tsuyoshi Yamamoto, and Tomoyuki Nishita. "An interactive rendering system using hierarchical data structure for earth-scale clouds". In: *Science China Information Sciences* 53.5 (2010), pp. 920–931.

[Ebe+02] David S Ebert, F Kenton Musgrave, Darwyn Peachey, Ken Perlin, and Steve Worley. *Texturing and modeling: a procedural approach*. Elsevier, 2002.

[EUT19] Marie-Lena Eckert, Kiwon Um, and Nils Thuerey. "Scalarflow: a large-scale volumetric data set of real-world scalar transport flows for computer animation and machine learning". In: *ACM Transactions on Graphics (TOG)* 38.6 (2019), pp. 1–16.

[FDY15] Charles Welton Ferreira Barbosa, Yoshinori Dobashi, and Tsuyoshi Yamamoto. "Adaptive cloud simulation using position based fluids". In: *Computer Animation and Virtual Worlds* 26.3-4 (2015), pp. 367–375.

[FL04] Raanan Fattal and Dani Lischinski. "Target-driven smoke animation". In: *ACM SIGGRAPH 2004 Papers*. 2004, pp. 441–448.

[FM97] Nick Foster and Dimitris Metaxas. "Modeling the motion of a hot, turbulent gas". In: *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. 1997, pp. 181–188.

[FSJ01] Ronald Fedkiw, Jos Stam, and Henrik Wann Jensen. "Visual simulation of smoke". In: *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. 2001, pp. 15–22.

[Gar85] Geoffrey Y Gardner. "Visual simulation of clouds". In: *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*. 1985, pp. 297–304.

[GBW97] Lori S Glaze, Stephen M Baloga, and Lionel Wilson. "Transport of atmospheric water vapor by volcanic eruption columns". In: *Journal of Geophysical Research: Atmospheres* 102.D5 (1997), pp. 6099–6108.

[Gir+23] Gilson Antonio Giraldi, Liliane Rodrigues de Almeida, Antonio Lopes Apolinário Jr, and Leandro Tavares da Silva. *Deep Learning for Fluid Simulation and Animation: Fundamentals, Modeling, and Case Studies*. Springer Nature, 2023.

[GJ+10] Gaël Guennebaud, Benoît Jacob, et al. *Eigen v3*. http://eigen.tuxfamily.org. 2010.

[GL94] JS Gilbert and SJ Lane. "The origin of accretionary lapilli". In: *Bulletin of Volcanology* 56 (1994), pp. 398–411.

[GLM17] Bruno Galerne, Arthur Leclaire, and Lionel Moisan. "Texton noise". In: *Computer Graphics Forum*. Vol. 36. 8. Wiley Online Library. 2017, pp. 205–218.

[GN17] Prashant Goswami and Fabrice Neyret. "Real-time landscape-size convective clouds simulation and rendering". In: *Workshop on Virtual Reality Interaction and Physical Simulation (VRIPHYS), Lyon*. Eurographics-European Association for Computer Graphics. 2017.

[Gos+23] Prashant Goswami, Abbas Cheddad, Fredrik Junede, and Samuel Asp. "Interactive landscape–scale cloud animation using DCGAN". In: *Frontiers in Computer Science* 5 (2023), p. 957920.

[Gos19] Prashant Goswami. "Interactive animation of single-layer cumulus clouds using cloudmap". In: *STAG2019–Smart Tools and Applications in Graphics, Cagliari, November 14-15*. Eurographics-European Association for Computer Graphics. 2019.

[Gos20] Prashant Goswami. "A survey of modeling, rendering and animation of clouds in computer graphics". In: *The Visual Computer* (2020), pp. 1–18.

[GZD08] Alexander Goldberg, Matthias Zwicker, and Frédo Durand. "Anisotropic noise". In: *ACM Transactions on Graphics (TOG)* 27.3 (2008), pp. 1–8.

[Häd+20] Torsten Hädrich, Miłosz Makowski, Wojciech Pałubicki, Daniel Banuti, Sören Pirk, and Dominik L Michels. "Stormscapes: Simulating Cloud Dynamics in the Now". In: *ACM Trans. Graph* 39.6 (2020).

[Häd+21] Torsten Hädrich, Daniel T Banuti, Wojtek Pałubicki, Sören Pirk, and Dominik L Michels. "Fire in paradise: Mesoscale simulation of wildfires". In: *ACM Transactions on Graphics (TOG)* 40.4 (2021), pp. 1–15.

[Har+03] Mark J Harris, William V Baxter, Thorsten Scheuermann, and Anselmo Lastra. "Simulation of cloud dynamics on graphics hardware". In: *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*. 2003, pp. 92–101.

[Hen83] Thomas H Henderson. "The Shuttle Mission Simulator Computer-Generated Imagery". In: (1983).

[Hin18] Jonathan Hines. "Stepping up to summit". In: *Computing in science & engineering* 20.2 (2018), pp. 78–82.

[Hof+23] Davis W Hoffman, Larry G Mastin, Alexa R Van Eaton, Stephen A Solovitz, Raul B Cal, and John K Eaton. "A One-Dimensional Volcanic Plume Model for Predicting Ash Aggregation". In: *Journal of Geophysical Research: Solid Earth* 128.9 (2023), e2023JB027002.

[HS+52] Magnus R Hestenes, Eduard Stiefel, et al. "Methods of conjugate gradients for solving linear systems". In: *Journal of research of the National Bureau of Standards* 49.6 (1952), pp. 409–436.

[HW65] Francis H Harlow and J Eddie Welch. "Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface". In: *The physics of fluids* 8.12 (1965), pp. 2182–2189.

[Jam+08] MR James, L Wilson, SJ Lane, JS Gilbert, TA Mather, RG Harrison, and RS Martin. "Electrical charging of volcanic plumes". In: *Space Science Reviews* 137 (2008), pp. 399–418.

[Jen+15] Susanna F Jenkins, TM Wilson, C Magill, V Miller, C Stewart, R Blong, W Marzocchi, Michael Boulton, Costanza Bonadonna, A Costa, et al. "Volcanic ash fall hazard and risk". In: *Global volcanic hazards and risk* (2015), pp. 173–222.

[Kal+17] Simon Kallweit, Thomas Müller, Brian Mcwilliams, Markus Gross, and Jan Novák. "Deep scattering: Rendering atmospheric clouds with radiance-predicting neural networks". In: *ACM Transactions on Graphics (TOG)* 36.6 (2017), pp. 1–11.

[KEM18] Janine L Kavanagh, Samantha L Engwell, and Simon A Martin. "A review of laboratory and numerical modelling in volcanology". In: *Solid Earth* 9.2 (2018), pp. 531–571.

[Kha+22] Sergey Khaykin, Aurelien Podglajen, Felix Ploeger, Jens-Uwe Grooß, Florent Tencé, Slimane Bekki, Konstantin Khlopenkov, Kristopher Bedka, Landon Rieger, Alexandre Baron, et al. "Global perturbation of stratospheric water and aerosol burden by Hunga eruption". In: *Communications Earth & Environment* 3.1 (2022), p. 316.

[Kim+19] Byungsoo Kim, Vinicius C Azevedo, Nils Thuerey, Theodore Kim, Markus Gross, and Barbara Solenthaler. "Deep fluids: A generative network for parameterized fluid simulations". In: *Computer graphics forum*. Vol. 38. 2. Wiley Online Library. 2019, pp. 59–70.

[KK14] Genichi Kawada and Takashi Kanai. "Controlling the Shape and Motion of Plumes in Explosion Simulations". In: *Proceedings of the 11th Workshop on Virtual Reality Interaction and Physical Simulation*. Vol. 2014. 2014, pp. 69–78.

[KV84] James T Kajiya and Brian P Von Herzen. "Ray tracing volume densities". In: *ACM SIGGRAPH computer graphics* 18.3 (1984), pp. 165–174.

[Lag+09] Ares Lagae, Sylvain Lefebvre, George Drettakis, and Philip Dutré. "Procedural noise using sparse Gabor convolution". In: *ACM Transactions on Graphics (TOG)* 28.3 (2009), pp. 1–10.

[Lag+10] Ares Lagae, Sylvain Lefebvre, Rob Cook, Tony DeRose, George Drettakis, David S Ebert, John P Lewis, Ken Perlin, and Matthias Zwicker. "A survey of procedural noise functions". In: *Computer Graphics Forum*. Vol. 29. 8. Wiley Online Library. 2010, pp. 2579–2600.

[Las+22] Maud Lastic, Damien Rohmer, Guillaume Cordonnier, Claude Jaupart, Fabrice Neyret, and Marie-Paule Cani. "Interactive simulation of plume and pyroclastic volcanic ejections". In: *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 5.1 (2022), pp. 1–15.

[Lea93] Doug Lea. "The GNU C++ library". In: *The C++ Report* (1993).

[Lew84] John-Peter Lewis. "Texture synthesis for digital painting". In: *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*. 1984, pp. 245–252.

[Lew89] John-Peter Lewis. "Algorithms for solid noise synthesis". In: *Proceedings of the 16th annual conference on Computer graphics and interactive techniques*. 1989, pp. 263–270.

[MB15] Ksenia Mukhina and Alexey Bezgodov. "The Method for Real-time Cloud Rendering". In: *Procedia Computer Science* 66 (2015), pp. 697–704.

[McK+21] Kathleen McKee, Cassandra M Smith, Kevin Reath, Eveanjelene Snee, Sean Maher, Robin S Matoza, Simon Carn, Larry Mastin, Kyle Anderson, David Damby, et al. "Evaluating the state-of-the-art in remote volcanic eruption characterization Part I: Raikoke volcano, Kuril Islands". In: *Journal of Volcanology and Geothermal Research* 419 (2021), p. 107354.

[MDN02a] Ryo Miyazaki, Yoshinori Dobashi, and Tomoyuki Nishita. "Simulation of Cumuliform Clouds Based on Computational Fluid Dynamics." In: *Eurographics (Short Presentations)*. 2002.

[MDN02b] Ryoichi Mizuno, Yoshinori Dobashi, and Tomoyuki Nishita. "Modeling of volcanic clouds using CML". In: *Proceedings of the International Computer Symposium*. 2002.

[Mén+20] Joshua Méndez Harper, Leah Courtland, Josef Dufek, and Julian McAdams. "Microphysical effects of water content and temperature on the triboelectrification of volcanic ash on long time scales". In: *Journal of Geophysical Research: Atmospheres* 125.14 (2020), e2019JD031498.

[Mil+22] Luis Millan, Michelle L Santee, Alyn Lambert, Nathaniel J Livesey, Frank Werner, Michael J Schwartz, Hugh Charles Pumphrey, Gloria L Manney, Yuan Wang, Hui Su, et al. "The Hunga Tonga-Hunga Ha'apai hydration of the stratosphere". In: *Geophysical Research Letters* 49.13 (2022), e2022GL099381.

[Miy+01] Ryo Miyazaki, Satoru Yoshida, Yoshinori Dobashi, and Tomoyuki Nishita. "A method for modeling clouds based on atmospheric fluid dynamics". In: *Proceedings Ninth Pacific Conference on Computer Graphics and Applications. Pacific Graphics 2001*. IEEE. 2001, pp. 363–372.

[Miz+03] Ryoichi Mizuno, Yoshinori Dobashi, Bing-Yu Chen, and Tomoyuki Nishita. "Physics motivated modeling of volcanic clouds as a two fluids model". In: *11th Pacific Conference on Computer Graphics and Applications, 2003. Proceedings.* IEEE. 2003, pp. 440–444.

[MM76] A.B. Meinel and M.P. Meinel. *Applied Solar Energy: An Introduction.* Addison-Wesley series in physics. Addison-Wesley Publishing Company, 1976. ISBN: 9780201047196. URL: `https://books.google.co.za/books?id=o9BSAAAAMAAJ`.

[MSB71] RL Mancuso, SM Serebreny, and RH Blackmer Jr. "A cloud simulation model for evaluating automatic ceilometer systems". In: *Journal of Applied Meteorology* 10.6 (1971), pp. 1324–1330.

[MTT56] Bruce R Morton, Geoffrey Ingram Taylor, and John Stewart Turner. "Turbulent gravitational convection from maintained and instantaneous sources". In: *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences* 234.1196 (1956), pp. 1–23.

[MV68] Benoit B Mandelbrot and John W Van Ness. "Fractional Brownian motions, fractional noises and applications". In: *SIAM review* 10.4 (1968), pp. 422–437.

[Ner+21] Augusto Neri, Tomaso Esposti Ongaro, Mattia de'Michieli Vitturi, and Matteo Cerminara. "Multiphase flow modeling of explosive volcanic eruptions". In: *Transport phenomena in multiphase systems*. Springer, 2021, pp. 243–281.

[Ney97] Fabrice Neyret. "Qualitative simulation of convective cloud formation and evolution". In: *Computer animation and simulation'97*. Springer, 1997, pp. 113–124.

[OA76] National Oceanic and Atmospheric Administration. "US standard atmosphere, 1976". In: *Technical Report* (1976).

[Obe+98] Josef M Oberhuber, Michael Herzog, Hans-F Graf, and Karsten Schwanke. "Volcanic plume simulation on large scales". In: *Journal of Volcanology and Geothermal Research* 87.1-4 (1998), pp. 29–53.

[Ola+02] Marc Olano, John Hart, Wolfgang Heidrich, and Michael McCool. *Real-Time Shading*. CRC Press, 2002.

[Ong+07] T Esposti Ongaro, Carlo Cavazzoni, Giovanni Erbacci, Augusto Neri, and Maria-Vittoria Salvetti. "A parallel multiphase flow code for the 3D simulation of explosive volcanic eruptions". In: *Parallel Computing* 33.7-8 (2007), pp. 541–560.

[Org17] World Meteorological Organization. *International Cloud Atlas*. World Meteorological Organization, 2017.

[Pał+22] Wojtek Pałubicki, Miłosz Makowski, Weronika Gajda, Torsten Hädrich, Dominik L Michels, and Sören Pirk. "Ecoclimates: Climate-response modeling of vegetation". In: *ACM Transactions on Graphics (TOG)* 41.4 (2022), pp. 1–19.

[Pat+05] Sanjit Patel, Anson Chu, Jonathan Cohen, and Frederic Pighin. "Fluid simulation via disjoint translating grids". In: *ACM SIGGRAPH 2005 Sketches*. 2005, 139–es.

[Per02] Ken Perlin. "Improving noise". In: *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*. 2002, pp. 681–682.

[Per85] Ken Perlin. "An image synthesizer". In: *ACM Siggraph Computer Graphics* 19.3 (1985), pp. 287–296.

[Pey+09] Adrien Peytavie, Eric Galin, Jérôme Grosjean, and Stéphane Mérillou. "Arches: a framework for modeling complex terrains". In: *Computer graphics forum*. Vol. 28. 2. Wiley Online Library. 2009, pp. 457–467.

[Pie+98] Roger A Pielke, RonI Avissar, Michael Raupach, A Johannes Dolman, Xubin Zeng, and A Scott Denning. "Interactions between the atmosphere and terrestrial ecosystems: influence on weather and climate". In: *Global change biology* 4.5 (1998), pp. 461–475.

[PN01] Ken Perlin and Fabrice Neyret. "Flow noise". In: *28th International Conference on Computer Graphics and Interactive Techniques (Technical Sketches and Applications)*. SIGGRAPH. 2001, p. 187.

[PPS22] Simon R Proud, Andrew T Prata, and Simeon Schmauß. "The January 2022 eruption of Hunga Tonga-Hunga Ha'apai volcano reached the mesosphere". In: *Science* 378.6619 (2022), pp. 554–557.

[PW09] Liz Parfitt and Lionel Wilson. *Fundamentals of physical volcanology*. John Wiley & Sons, 2009.

[Pyl89] David M Pyle. "The thickness, volume and grainsize of tephra fall deposits". In: *Bulletin of Volcanology* 51 (1989), pp. 1–15.

[Rav+21] Suman Ravuri, Karel Lenc, Matthew Willson, Dmitry Kangin, Remi Lam, Piotr Mirowski, Megan Fitzsimons, Maria Athanassiadou, Sheleem Kashem, Sam Madge, et al. "Skilful precipitation nowcasting using deep generative models of radar". In: *Nature* 597.7878 (2021), pp. 672–677.

[RD11] William I Rose and Adam J Durant. "Fate of volcanic ash: Aggregation and fallout". In: *Geology* 39.9 (2011), pp. 895–896.

[RLB19] John R. Rumble, David R. Lide, and Thomas J. Bruno. *CRC handbook of chemistry and physics : a ready-reference book of chemical and physical data*. eng. 2019-2020, 100th edition. Boca Raton, FL: CRC Press, 2019. ISBN: 113836729X.

[RW94] Todd Reed and Brian Wyvill. "Visual simulation of lightning". In: *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*. 1994, pp. 359–364.

[Sat+22] Pinar Satilmis, Demetris Marnerides, Kurt Debattista, and Thomas Bashford-Rogers. "Deep synthesis of cloud lighting". In: *IEEE Computer Graphics and Applications* 42.5 (2022), pp. 8–18.

[SB08] Hagit Schechter and Robert Bridson. "Evolving sub-grid turbulence for smoke animation". In: *Proceedings of the 2008 ACM SIGGRAPH/Eurographics symposium on Computer animation*. 2008, pp. 1–7.

[Sch+03] Joshua Schpok, Joseph Simons, David S Ebert, and Charles Hansen. "A real-time cloud modeling, rendering, and animation system". In: *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*. Citeseer. 2003, pp. 160–166.

[Sch29] Erich Schoenberg. *Theoretische Photometrie*. Springer, 1929.

[Sha+04] Maurya Shah, Jonathan M Cohen, Sanjit Patel, Penne Lee, and Frédéric Pighin. "Extended galilean invariance for adaptive fluid simulation". In: *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*. 2004, pp. 213–221.

[SK10] YJ Suzuki and T Koyaguchi. "Numerical determination of the efficiency of entrainment in volcanic eruption columns". In: *Geophysical Research Letters* 37.5 (2010).

[SM20] Microsoft Flight Simulator and Mathias Müller. *Partnership Series: Meteoblue - Weather Forecast System*. Microsoft Flight Simulator and Meteoblue. June 2020. URL: https://web.archive.org/web/20200612005006/https://www.youtube.com/watch?v=3AfQGA8i7uQ.

[Smi+21] Cassandra M Smith, Damien Gaudin, Alexa R Van Eaton, Sonja A Behnke, Steven Reader, Ronald J Thomas, Harald Edens, Stephen R McNutt, and Corrado Cimarelli. "Impulsive volcanic plumes generate volcanic lightning and vent discharges: A statistical analysis of Sakurajima volcano in 2015". In: *Geophysical Research Letters* 48.11 (2021), e2020GL092323.

[SN01] Christopher Small and Terry Naumann. "The global distribution of human population and recent volcanism". In: *Global Environmental Change Part B: Environmental Hazards* 3.3 (2001), pp. 93–109.

[Søn+20] Casper Kaae Sønderby, Lasse Espeholt, Jonathan Heek, Mostafa Dehghani, Avital Oliver, Tim Salimans, Shreya Agrawal, Jason Hickey, and Nal Kalchbrenner. "Metnet: A neural weather model for precipitation forecasting". In: *arXiv preprint arXiv:2003.12140* (2020).

[Sta99] Jos Stam. "Stable Fluids". In: *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '99. USA: ACM Press/Addison-Wesley Publishing Co., 1999, pp. 121–128. ISBN: 0201485605. DOI: 10.1145/311535.311548. URL: https://doi.org/10.1145/311535.311548.

[Ste+19] S Stern, Corrado Cimarelli, D Gaudin, B Scheu, and DB Dingwell. "Electrification of experimental volcanic jets with varying water content and temperature". In: *Geophysical Research Letters* 46.20 (2019), pp. 11136–11145.

[Sto+99] Dan Stora, Pierre-Olivier Agliati, Marie-Paule Cani, Fabrice Neyret, and Jean-Dominique Gascuel. "Animating lava flows". In: *Graphics Interface (GI'99) Proceedings*. 1999, pp. 203–210.

[Suz+16] YJ Suzuki, Antonio Costa, Matteo Cerminara, T Esposti Ongaro, Michael Herzog, Alexa R Van Eaton, and LC Denby. "Inter-comparison of three-dimensional models of volcanic plumes". In: *Journal of Volcanology and Geothermal Research* 326 (2016), pp. 26–42.

[SY05] Lin Shi and Yizhou Yu. "Controllable smoke animation with guiding objects". In: *ACM Transactions on Graphics (TOG)* 24.1 (2005), pp. 140–164.

[Tet30] Otto Tetens. "Uber einige meteorologische Begriffe". In: *Z. Geophys* 6 (1930), pp. 297–309.

[Tex+05] Christiane Textor, Hans F Graf, Antonella Longo, Augusto Neri, T Esposti Ongaro, Paolo Papale, Claudia Timmreck, and GG Ernst. "Numerical simulation of explosive volcanic eruptions from the conduit flow to global atmospheric scales". In: *Annals of Geophysics* 48 (2005), pp. 817–842.

[TKP13] Nils Thuerey, Theodore Kim, and Tobias Pfaff. "Turbulent fluids". In: *ACM SIGGRAPH 2013 Courses*. 2013, pp. 1–1.

[Tre+03] Adrien Treuille, Antoine McNamara, Zoran Popović, and Jos Stam. "Keyframe control of smoke simulations". In: *ACM SIGGRAPH 2003 Papers*. 2003, pp. 716–723.

[Tri+19] Thibault Tricard, Semyon Efremov, Cédric Zanni, Fabrice Neyret, Jonàs Martínez, and Sylvain Lefebvre. "Procedural phasor noise". In: *ACM Transactions on Graphics (TOG)* 38.4 (2019), pp. 1–13.

[Van+15] Alexa R Van Eaton, Larry G Mastin, Michael Herzog, Hans F Schwaiger, David J Schneider, Kristi L Wallace, and Amanda B Clarke. "Hail formation triggers rapid ash aggregation in volcanic plumes". In: *Nature communications* 6.1 (2015), p. 7860.

[Van+16] Alexa R Van Eaton, Álvaro Amigo, Daniel Bertin, Larry G Mastin, Raúl E Giacosa, Jerónimo González, Oscar Valderrama, Karen Fontijn, and Sonja A Behnke. "Volcanic lightning and plume behavior reveal evolving hazards during the April 2015 eruption of Calbuco volcano, Chile". In: *Geophysical Research Letters* 43.7 (2016), pp. 3563–3571.

[Van91] Jarke J Van Wijk. "Spot noise texture synthesis for data visualization". In: *Proceedings of the 18th annual conference on Computer graphics and interactive techniques*. 1991, pp. 309–318.

[Vim+20] Ulysse Vimont, James Gain, Maud Lastic, Guillaume Cordonnier, Babatunde Abiodun, and Marie-Paule Cani. "Interactive Meso-scale Simulation of Skyscapes". In: *Computer Graphics Forum*. Vol. 39. 2. 2020.

[Voh+99] O Vohl, SK Mitra, SC Wurzler, and HR Pruppacher. "A wind tunnel study of the effects of turbulence on the growth of cloud drops by collision and coalescence". In: *Journal of the atmospheric sciences* 56.24 (1999), pp. 4088–4099.

[WB07] Roy Walmsley and Brian Butters. "Infrared smoke modelling in CounterSim". In: *Technologies for Optical Countermeasures IV*. Vol. 6738. SPIE. 2007, pp. 215–223.

[Web+18] Antoine Webanck, Yann Cortial, Eric Guérin, and Eric Galin. "Procedural cloudscapes". In: *Computer Graphics Forum*. Vol. 37. 2. Wiley Online Library. 2018, pp. 431–442.

[Wed+20] Nils P Wedi, Inna Polichtchouk, Peter Dueben, Valentine G Anantharaj, Peter Bauer, Souhail Boussetta, Philip Browne, Willem Deconinck, Wayne Gaudin, Ioan Hadade, et al. "A baseline for global weather and climate simulations at 1 km resolution". In: *Journal of Advances in Modeling Earth Systems* 12.11 (2020), e2020MS002192.

[Wer+19] Maximilian Werhahn, You Xie, Mengyu Chu, and Nils Thuerey. "A multi-pass GAN for fluid flow super-resolution". In: *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 2.2 (2019), pp. 1–21.

[WH06] John M Wallace and Peter V Hobbs. *Atmospheric science: an introductory survey*. Vol. 92. Elsevier, 2006.

[Wil+21] Alexander Wilkie, Petr Vevoda, Thomas Bashford-Rogers, Lukáš Hošek, Tomáš Iser, Monika Kolářová, Tobias Rittig, and Jaroslav Křivánek. "A fitted radiance and attenuation model for realistic atmospheres". In: *ACM Transactions on Graphics (TOG)* 40.4 (2021), pp. 1–14.

[WK94] Andrew W Woods and Juergen Kienle. "The dynamics and thermodynamics of volcanic clouds: theory and observations from the April 15 and April 21, 1990 eruptions of Redoubt Volcano, Alaska". In: *Journal of Volcanology and Geothermal Research* 62.1-4 (1994), pp. 273–299.

[WM81] Richard T Wetherald and Syukuro Manabe. "Simulations of Cloud Cover with a Global General Circulation Model of the Atmosphere". In: *Clouds in Climate: Modeling and Satellite Observational Studies* (1981), pp. 27–33.

[Wor96] Steven Worley. "A cellular texture basis function". In: *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. 1996, pp. 291–294.

[Wri+22] Corwin J Wright, Neil P Hindley, M Joan Alexander, Mathew Barlow, Lars Hoffmann, Cathryn N Mitchell, Fred Prata, Marie Bouillon, Justin Carstens, Cathy Clerbaux, et al. "Surface-to-space atmospheric waves from Hunga Tonga–Hunga Ha'apai eruption". In: *Nature* 609.7928 (2022), pp. 741–746.

[Xie+18] You Xie, Erik Franz, Mengyu Chu, and Nils Thuerey. "tempoGAN: A temporally coherent, volumetric GAN for super-resolution fluid flow". In: *ACM Transactions on Graphics (TOG)* 37.4 (2018), pp. 1–15.

[Xu+09a] Jiangbin Xu, Hanchen Song, Lingda Wu, and Chao Yang. "A seamless representation of multi-scale clouds". In: *2009 Fifth International Joint Conference on INC, IMS and IDC*. IEEE. 2009, pp. 1255–1260.

[Xu+09b] Jiangbin Xu, Chao Yang, Jian Zhao, and Lingda Wu. "Fast modeling of realistic clouds". In: *2009 International Symposium on Computer Network and Multimedia Technology*. IEEE. 2009, pp. 1–4.

[YI07] Akihiko Yokoo and Kazuhiro Ishihara. "Analysis of pressure waves observed in Sakurajima eruption movies". In: *Earth, planets and space* 59 (2007), pp. 177–181.

[YOH00] Gary D Yngve, James F O'Brien, and Jessica K Hodgins. "Animating explosions". In: *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. 2000, pp. 29–36.

[Yua+14] Chunqiang Yuan, Xiaohui Liang, Shiyu Hao, Yue Qi, and Qinping Zhao. "Modelling cumulus cloud shape from a single image". In: *Computer Graphics Forum*. Vol. 33. 6. Wiley Online Library. 2014, pp. 288–297.

[Zar20] Omar Zarifi. "Sparse smoke simulations in Houdini". In: *Special Interest Group on Computer Graphics and Interactive Techniques Conference Talks*. 2020, pp. 1–2.

[Zha+17] Shenfan Zhang, Fanlong Kong, Chen Li, Changbo Wang, and Hong Qin. "Hybrid modeling of multiphysical processes for particle-based volcano animation". In: *Computer Animation and Virtual Worlds* 28.3-4 (2017), e1758.

Appendices

Appendix A

2D pressure solve when $\Delta x \neq \Delta y$

$$\frac{u_{i,\,j+1/2} - u_{i,\,j-1/2}}{\Delta x} + \frac{v_{i+1/2,\,j} - v_{i-1/2,\,j}}{\Delta y} = 0$$

$$\frac{\left(u_{i,\,j+1/2} - \frac{\Delta t}{\rho}\frac{p_{i,\,j+1} - p_{i,\,j}}{\Delta x}\right) - \left(u_{i,\,j-1/2} - \frac{\Delta t}{\rho}\frac{p_{i,\,j} - p_{i,\,j-1}}{\Delta x}\right)}{\Delta x} +$$

$$\frac{\left(v_{i+1/2,\,j} - \frac{\Delta y}{\rho}\frac{p_{i+1,\,j} - p_{i,\,j}}{\Delta y}\right) - \left(v_{i-1/2,\,j} - \frac{\Delta t}{\rho}\frac{p_{i,\,j} - p_{i-1,\,j}}{\Delta y}\right)}{\Delta y} = 0$$

$$\frac{u_{i,\,j+1/2} - u_{i,\,j-1/2} + \frac{\Delta t}{\rho \Delta x}\left(2p_{i,\,j} - p_{i,\,j+1} - p_{i,\,j-1}\right)}{\Delta x} +$$

$$\frac{v_{i+1/2,\,j} - u_{i-1/2,\,j} + \frac{\Delta t}{\rho \Delta x}\left(2p_{i,\,j} - p_{i+1,\,j} - p_{i-1,\,j}\right)}{\Delta y} = 0$$

$$\frac{\Delta t}{\rho}\left(\left(\frac{2x}{y} + \frac{2y}{x}\right)p_{i,\,j} - \frac{y}{x}p_{i,\,j+1} - \frac{y}{x}p_{i,\,j-1} - \frac{x}{y}p_{i+1,\,j} - \frac{x}{y}p_{i-1,\,j}\right) =$$

$$\Delta y \cdot u_{i,\,j-1/2} - \Delta y \cdot u_{i,\,j+1/2} + \Delta x \cdot v_{i-1/2,\,j} - \Delta x \cdot v_{i+1/2,\,j}$$

Appendix B

Derivation of solid interface in pressure solve

Let's assume that cells $i,\ j-1$ and $i-1,\ j$ is solid, and that $\Delta x \equiv \Delta y$. We assume that the solid interfaces are stationary (since mountains don't walk) and thus $u_S = 0$. Therefore, the final pressure calculation for cell $i,\ j$ will be

$$\frac{u_{i,\,j+1/2} - u_{i,\,j-1/2} + v_{i+1/2,\,j} - v_{i-1/2,\,j}}{\Delta x} = 0$$

$$\frac{u_{i,\,j+1/2} - u_S + v_{i+1/2,\,j} - u_S}{\Delta x} = 0$$

$$\frac{\left(u_{i,\,j+1/2} - \frac{\Delta t}{\rho}\frac{p_{i,\,j+1} - p_{i,\,j}}{\Delta x}\right) - 0}{\Delta x} + \frac{\left(v_{i+1/2,\,j} - \frac{\Delta t}{\rho}\frac{p_{i+1,\,j} - p_{i,\,j}}{\Delta x}\right) - 0}{\Delta x} = 0$$

$$\frac{\Delta t}{\rho}\frac{2p_{i,\,j} - p_{i+1,\,j} - p_{i,\,j+1}}{\Delta x^2} = -(\frac{u_S - u_{i+1/2,\,j}}{\Delta x} + \frac{v_{i,\,j+1/2} - u_S}{\Delta x})$$

Appendix C

Example calculation of layer heights

The user provides a list of heights (in metres), a maximum altitude, a maximum number of layers, and a maximum layer thickness. More layers require more computation, but thicker layers lead to less fidelity in the output. In this example, we use $[500, 1500, 2500, 2700, 3000, 3500, 4500, 5500, 6500]$ as the provided list, with a maximum altitude of 20000 metres, a maximum layer count of 20, and a maximum layer thickness of 1000 metres.

The initial layers (centred around the altitudes of interest to the user) are generated in a three phase process. Table C.1 demonstrates the values in practice. The first phase iterates through each height h, and finds the distance d to the nearest neighbour in the list. Initial bounds of $[h - d/2, h + d/2]$ are created. This might result in gaps between layers. The second phase iterates through the newly created list of bounds and expands each layer's bounds until the layer is in contact with at least one neighbour. The bounds are sorted with the thickest layer first before the final phase. If a layer is not in contact with both neighbours, it is expanded until all gaps are removed. The sorting is required to ensure that the loss in fidelity is minimised, and areas of interest with narrow separation (e.g., 2500, 2700, and 3000 in this example) maintain that narrow separation.

If, after the third phase, a layer exceeds the specified maximum thickness, it is split evenly to form two layers. In our example, that means the layer with bounds $[750, 2400]$ is split into $[750, 1575]$ and $[1575, 2400]$, and $[3850, 5000]$ goes to $[3850, 4425]$ and $[4425, 5000]$. Finally, the space between the highest layer generated ($[6000, 7000]$ in our example) and the maximum altitude is evenly divided into layers such that the total number of layers is equal to the maximum number of layers. That is, with 11 layers already generated, the remaining 13000m is divided into 9 layers each 1444.4m thick. Table C.2 provides the reference heights and final bounds for all layers, with the provided list $[500, 1500, 2500, 2700, 3000, 3500, 4500, 5500, 6500]$, a maximum thickness of 1000m, a maximum altitude of 20000m, and 20 as the maximum number of layers.

User height (m)	**After phase 1**	**After phase 2**	**After phase 3**
500	$[250, 750]$	$[250, 750]$	$[0, 750]$
1500	$[1000, 2000]$	$[750, 2250]$	$[750, 2400]$
2500	$[2400, 2600]$	$[2400, 2600]$	$[2400, 2600]$
2700	$[2600, 2800]$	$[2600, 2800]$	$[2600, 2800]$
3000	$[2850, 3150]$	$[2850, 3150]$	$[2800, 3150]$
3500	$[3250, 3750]$	$[3150, 3850]$	$[3150, 3850]$
4500	$[4000, 5000]$	$[4000, 5000]$	$[3850, 5000]$
5500	$[5000, 6000]$	$[5000, 6000]$	$[5000, 6000]$
6500	$[6000, 7000]$	$[6000, 7000]$	$[6000, 7000]$

Table C.1: The bounds of the air layers generated by the user-provided list of altitudes after each phase of the initialisation, where the list provided was $[500, 1500, 2500, 2700, 3000, 3500, 4500, 5500, 6500]$.

Reference height (m)	**Bounds**
375	$[0, 750]$
1162.5	$[750, 1575]$
1987.5	$[1575, 2400]$
2500	$[2400, 2600]$
2700	$[2600, 2800]$
2975	$[2800, 3150]$
3500	$[3150, 3850]$
4137.5	$[3850, 4425]$
4712.5	$[4425, 5000]$
5500	$[5000, 6000]$
6500	$[6000, 7000]$
7722.2	$[7000, 8444]$
9166.6	$[8444, 9889]$
10611.1	$[9889, 11333]$
12055.5	$[11333, 12778]$
13500	$[12778, 14222]$
14944.4	$[14222, 15666]$
16388.8	$[15666, 17111]$
17833.3	$[17111, 18555]$
19277.7	$[18555, 20000]$

Table C.2: The final layer heights and bounds after initialisation. The layers below 7000m are generated from the user-provided list $[500, 1500, 2500, 2700, 3000, 3500, 4500, 5500, 6500]$, with the layers above 7000m evenly divided up to the maximum altitude (20000m) and maximum number of layers (20).