Efficient visual simulation of volcanic phenomena

Maud Lastic

INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2023IPPAX049

Thèse de doctorat

Efficient visual simulation of volcanic phenomena

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à l'École polytechnique

École doctorale n°626 École doctorale de l'Institut Polytechnique de Paris (EDIPP)
Spécialité de doctorat : Informatique

Thèse présentée et soutenue à Palaiseau, le 3 juillet 2023, par

MAUD LASTIC

Composition du Jury :

Maud Marchal Professeure des universités, Université de Rennes, INSA (IRISA, UMR 6074)	Présidente
Sylvain Lefebvre Directeur de recherche, Inria, Université de Lorraine (LORIA, UMR 7503)	Rapporteur
Eric Paquette Professeur, Ecole de technologie supérieure de Montréal	Rapporteur
Kiwon Um Maître de conférences, Télécom Paris (LTCI, UMR 5141)	Examinateur
Claude Jaupart Professeur des universités, Université Paris Cité (IPGP, UMR 7154)	Examinateur
Marie-Paule Cani Professeure des universités, Ecole Polytechnique (LIX, UMR 7161)	Directrice de thèse
Damien Rohmer Professeur, Ecole Polytechnique (LIX, UMR 7161)	Co-encadrant de thèse

Résumé

L'un des principaux défis en informatique graphique est de permettre la création de mondes virtuels vastes et réalistes. Pour les enrichir, il faut pouvoir offrir des outils pour créer des phénomènes naturels cohérents divers et variés facilement et efficacement. Ainsi, des effets visuellement plausibles mais contrôlables donnent plus de liberté pour la création de films 3D, de simulations et de jeux vidéo.

Le but de cette thèse est de proposer de nouveaux modèles d'animation pour les éruptions volcaniques. Ces modèles sont pensés pour être utilisés par des artistes, posant certains objectifs à atteindre : être plausibles, c'est-à-dire avoir une ressemblance avec la réalité et une dynamique correcte ; être rapides, pour pouvoir être utilisés de manière interactive et dans les jeux vidéo ; et être contrôlables, en étant légers et simples à comprendre, pour que les artistes puissent les adapter à leurs besoins.

Pour animer des éruptions explosives, nous proposons un modèle prenant en compte leur dynamique, simulant des panaches montant à haute altitude et des coulées pyroclastiques dévalant les flancs du volcan, selon les conditions initiales. En effet, la colonne éruptive présente un caractère convectif et turbulent, ce qui entraîne l'air environnant et le réchauffe, faisant baisser sa densité. Si la densité du mélange de gaz et de cendres diminue assez et devient inférieure à celle de l'atmosphère, la poussée d'Archimède fait s'élever le panache haut dans l'atmosphère ; en revanche, si sa vitesse atteint zéro avant, elle retombe pour former des nuées ardentes. Notre modèle comporte deux couches : une simulation Lagrangienne minimaliste représentant des tranches horizontales de panache interagissant avec l'air environnant ; et un modèle procédural renforçant l'animation des flux turbulents avec des détails multi-résolution sous forme de sphères de plusieurs échelles de taille pour reproduire un aspect fractal. Nous étendons ce modèle en le combinant à un modèle atmosphérique où plusieurs couches horizontales représentent l'atmosphère, où nous simulons les phénomènes physiques menant à la formation des nuages. Ainsi, différents nuages peuvent être animés et influencés par la présence d'un panache.

Enfin, les coulées de lave font partie des phénomènes naturels les plus complexes, puisque la lave est un fluide non newtonien à la viscosité variable et aux comportements multiples en se refroidissant, passant par des états liquide, plastique et solide. L'aspect visuel de la lave est souvent hybride, avec des parties liquides sur lesquelles flottent des éléments plus froids, des croûtes déformables qui se plissent

et se déforment, et des parties rigides qui s'agrègent et sont portées par la coulée. De plus, la coulée, en refroidissant et en se solidifiant, modifie la surface du terrain sur laquelle elle s'écoule, et peut donc changer de trajectoire dans le temps. Des méthodes partielles de génération de coulées de lave ont été proposées en informatique graphique, ne prenant en compte que des sous-parties du problème, comme synthétiser une texture dont l'aspect change en étant transportée par la coulée. Aucune méthode n'a pu gérer l'interaction entre l'aspect visuel de la lave et la coulée, ni la formation et le plissement de surfaces déformables. Plutôt que de s'intéresser à de la simulation pure, nous proposons un modèle à couches combinant une simulation Eulérienne de lave et une simulation géométrique de la surface de la coulée, permettant l'apparition de plis en fonction de la vitesse et de la température. La simulation de lave décrit non seulement l'évolution spatio-temporelle de la coulée, mais également l'état dans lequel la lave se trouve, et notamment l'épaisseur de croûte solide à sa surface. Nous simulons également l'évolution des plis formés, de leur transport par la coulée fluide en-dessous à leurs collisions. Nous habillons la surface de la coulée avec des textures procédurales basées sur l'épaisseur de la croûte en respectant une cohérence temporelle.

Abstract

Offering tools to easily and efficiently create consistent natural phenomena is one of the main challenges in Computer Graphics, where visually plausible but controllable virtual effects are mandatory for 3D films, simulators and games.

The goal of this PhD is to propose novel multi-scale models for animating volcanic eruptions. These models are meant to be used by artists, giving several goals to achieve: being plausible, which means having a geometric resemblance to reality, as well as having the correct temporal dynamic; being fast in order to be able to be used interactively while permitting an usage in video games; and finally being controllable, with light and easy to understand model, so that users can easily adapt them to their needs.

To animate explosive eruptions, we propose a model that takes their unique dynamics into account, resulting into ascending plumes propagating upward and finally spreading side-way as well as pyroclastic flows spreading down the slopes of the volcano, depending on initial conditions. Our model combines two consistently coupled, simple sub-models: a minimalist Lagrangian simulation, used to represent dynamic horizontal slices of material ejected by the volcano and interacting with the surrounding air; and a procedural model that enhances the visual animation of the turbulent flow with multi-resolution details. We extend this model by combining it with an atmospheric model, where several horizontal layers represent the atmosphere and we simulate the physical phenomena leading to the formation of clouds. Thus, several types of clouds can be animated and interact with a plume.

Lastly, lava flows are among the most complex targeted phenomena, since they involve fluids evolving to a variety of behaviors while cooling down, from liquid to plastic, and then to rigid states. The visual aspect of lava is often hybrid, with liquid parts carrying cooler elements, and deformable crust that folds and deforms. None of these methods was able to handle the interaction between the visual state of the lava and the underlying flow, nor the formation and folding of deformable surface sheets. Therefore, rather than tackling pure simulation, we use as a base an existing Eulerian simulation of lava flows and build a geometrical simulation of the surface of the flow, letting folds appear knowing the velocity and temperature of the flow. We texture the flow using time-consistent textures generated according to the thickness of the crust.

Thanks

I first would like to thank my supervisors, Marie-Paule Cani and Damien Rohmer, for your advice and support throughout my PhD. Thank you Marie-Paule for having transmitted to me your love for the modeling of natural phenomena. Thank you Damien for your availability and your help when I had coding issues.

I would also like to thank the members of my jury for having accepted to be part of it. In particular, thank you to Eric Paquette and Sylvain Lefebvre who reviewed my thesis and for their helpful comments.

The work that I conducted during my PhD would not have been possible without my collaborators. A big thank you to Claude Jaupart for having taught me so much about volcanology. Thank you Eric Chahi and François Sahy for having given me access to your Volcano Simulator code and for your interest in my work. Thank you also to Guillaume Cordonnier, James Gain, Cilliers Pretorius, Pascal Guehl, Ulysse Vimont, Fabrice Neyret, Jean-Michel Dischler and Jiong Chen for your work, feedback and advice.

I also want to thank all the people from the Geovic/Vista/Geomerix teams I shared time with during my PhD, for the discussions at lunch or at a coffee break. Thank you Pooran, Vicky, Mathieu, Maks, Steve, Renaud, Jiayi, Jiong, Pascal, David-Henri, Tara, Tim, Eduardo, Ariel, Magali, Pauline, Thomas, Amal, Marie-Julie, and those I forget. Thank you Pierre for all our discussions. Thank you Nicolas for our mutual support throughout our PhD.

Merci également à mes ami·es pour leur présence durant ma thèse. Un énorme merci aux danseur·euses de Hinodemai avec qui j'ai partagé tant de joie et d'énergie pendant ces années, en espérant longtemps continuer à danser avec vous.

Merci à ma famille pour les moments reposants passés avec vous quand je vous rends visite. Merci à mes parents et à ma soeur.

Enfin, merci Jean-Côme de m'accompagner dans la vie.

Contents

Chapter 1

Introduction

Figure 1.1: Top left: the volcano divinity Te Ka in the animation movie Moana ©Disney. Top right: the volcano in the movie The Lord of the Rings: The Return of the King ©New Line Productions. Bottom left: a volcano in the video game From Dust ©Ubisoft. Bottom right: a volcano simulator in the museum La Cité du Volcan at La Réunion ©E.Chahi, F.Sahy.

1.1 Motivation

Volcanic eruptions are among the most impressive and fascinating natural phenomena that can occur on Earth, as they are strikingly beautiful but can have dramatic consequences on whole populations' lives. This is certainly why we find them so often in legends and stories, represented nowadays in movies and video games. Studying volcanoes and reproducing their behavior can have different objectives: while understanding them better to anticipate eruptions and protect local populations is important, we can use this knowledge to visually simulate them in a plausible way. Modeling and animating virtual

volcanoes can help to create special effects in video games (1.1-bottom left) and movies (1.1-top), and serve an educational purpose in museums (1.1-bottom right).

Even considering their interest, only little research in Computer Graphics has been conducted about volcanic eruptions. Indeed, plausible visual simulation of volcanic eruptions is a whole challenge. First, as volcanic eruptions are quite rare and dangerous, we lack data about them, and they are complicated to observe, study and reproduce. And second, on a more technical side, they are complex phenomena, and it is harder to simulate them than other fluids: the mixture of ash and gas ejected from volcanoes is more complex than a regular smoke because of its interaction with the surrounding air, and lava is more complex than other fluids because of its varying viscosity and it can go from a liquid to a solid state, with a hybrid appearance. Models used in geoscience to simulate these phenomena are quite heavy and it can take days to output a single simulation. Using such approaches in Computer Graphics would result in heavy models, hard to set up and parametrize, and that do not allow interactive control easily for potential users wishing to generate their own volcanic eruptions in their favorite setting. Indeed, artists want to be able to create eruptions for movies or games; and volcanologists could author eruptions to explain volcanology to the public. More specifically, volcanologists would like to able to use an interactive system letting them evaluate the consequences of eruptions with different initial conditions, and show them to local authorities and populations. We could also use learning-based methods that have become more and more popular throughout the last decade and are very accurate, but training them uses once again a lot of computation resources, while we are at a time of ecological crisis where we need to reduce our energy consumption.

Therefore, I wish to focus on lighter models in my work. Low energy cost is not the only advantage of such approaches: making a model lighter increases its efficiency, and helps make it real-time, permitting interactive user interaction and applications in video games. Furthermore, I wish to make models lighter, not only on the computational aspect but also by making them easy to understand and adapt if needed. For these reasons, I use multi-layer models in my research. In the animation of natural phenomena such as volcanic eruption, they consist in separating the phenomena into several sub-phenomena and treating each one in a different layer of the model.

1.2 Volcanic eruptions

There are several types of volcanoes, depending on how they form and where on the surface of the Earth. The inside of the Earth can be described as divided into three main parts (see also 1.2, from the inside to the surface:

- The core, divided into a solid inner core and a liquid outer core;

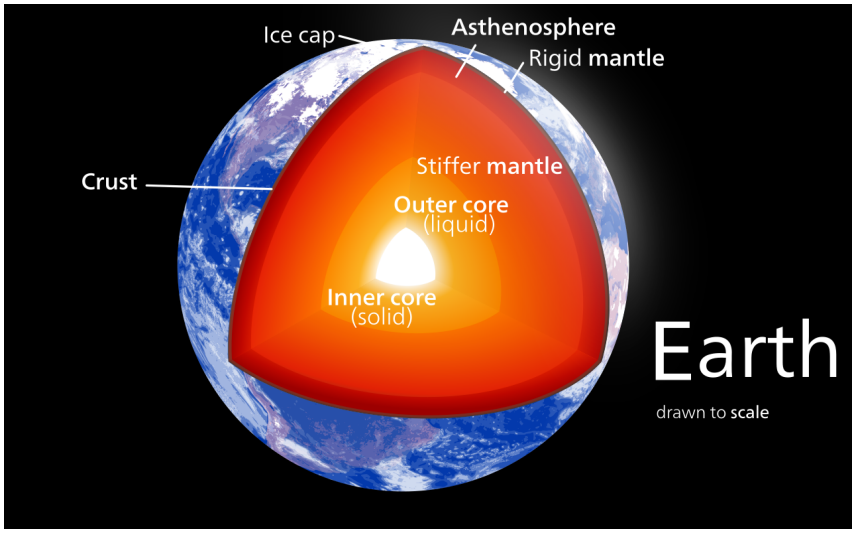

Figure 1.2: The different layers inside the Earth.

- The mantle, which is quite viscous, and where convection happens because of cool mantle near the surface going down to the core, causing movement of the tectonic plates;

- The lithosphere, divided into an elastic upper mantle layer and the solid crust, which is the thin surface of the Earth apparent to us. The crust is divided into tectonic plates that float on the upper mantle and can move due to the convection in the mantle.

The motion of the tectonic plates is one cause of the formation of volcanoes (see 1.3) Indeed, the fact that plates can move implies that they move away from each other in some places, like in the middle of the Atlantic Ocean, and that some have to go under other ones, for instance, next to Japan or on the Pacific coast of America.

Tectonic plates are generated by mantle upwellings beneath mid-ocean ridges, where magma formation is due to decompression melting. Such a magma is driven to the surface by buoyancy, forming whole lines of volcanoes. It happens mostly underwater, creating submarine volcanoes that may go up to the surface of the ocean at some point.

On the contrary, if two tectonic plates converge, one of them has to go beneath the other one: this is called subduction and causes several geological phenomena such as earthquakes and volcanoes. Indeed, water goes with the subducting plate and lowers the melting temperature of the mantle and the crust, thus forming magma that can sometimes reach the surface and form a volcano. When this happens in an ocean, it leads to the formation of volcanic islands like Japan or the Antilles. When this occurs on a continent, it forms a mountain range like the Andes in South America.

Another cause of volcano formation is hotspots. Indeed, some volcanoes appear far away from tectonic plate boundaries. It is due to hot columns in the mantle, causing magma to pierce through the

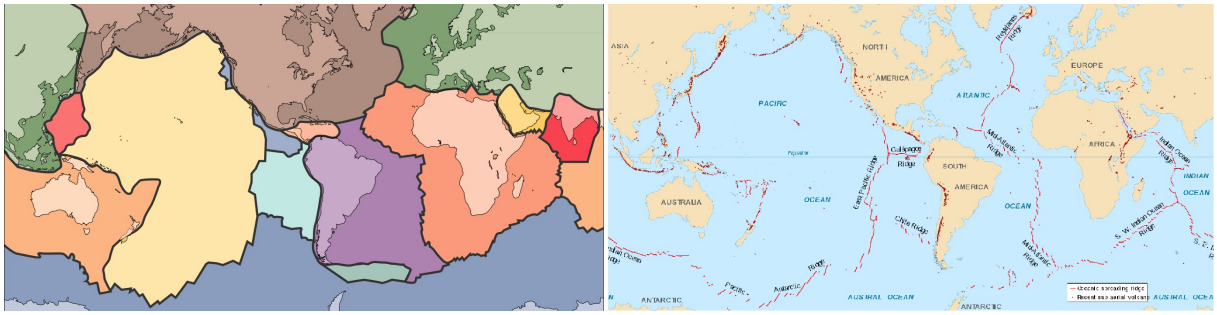

Figure 1.3: Left: Tectonic plates at the surface of the Earth. Right: Repartition of the volcanoes on Earth and spreading ridges. We can observe whole ranges of volcanoes where subduction occurs, as well as lonely volcanoes like Hawaii in the middle of the Pacific Ocean.

crust and emerge, thus forming volcanoes. When the tectonic plate where the spot is located moves, the hotspot underneath stays at the same place, and the place where it goes out as a volcano on the tectonic plate moves. It leads to the formation of ranges of volcanoes, like Hawaii (see 1.4) or Yellowstone.

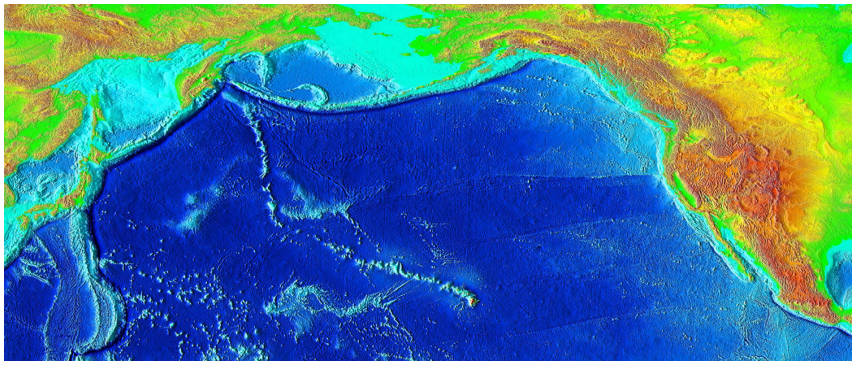

Figure 1.4: The trail formed by former and present volcanoes formed by the Hawaii hotspot while the Pacific tectonic plate was moving for millions of years. Only the most recent ones are high enough to form islands, the other ones are underwater. The most recent ones are also the only active volcanoes nowadays since the hotspot is underneath them.

The origin of the volcano partly explains what kind of eruptions are to be expected. Indeed, the type of eruption depends on the amount of gas and the viscosity of the lava, which depends on many factors, like the amount the silicon, the amount of water, and the type of rocks involved. These parameters vary depending whether the volcano results from a hotspot or from subduction, which leads to different types of eruptions with different objects that can be ejected from the volcano.

Before the eruption of a volcano, magma is stored in magma chambers located between 1 and 10 km below the surface. When large amounts of magma are stored close to the surface, it leads to bigger and faster eruptions. When the magma goes up toward the surface, its pressure greatly decreases. Because of that, the gas is no longer soluble in the magma: it forms small bubbles in the magma. If there is a lot of gas, it can cause all these bubbles to burst and smash the liquid magma. This phenomenon is called fragmentation.

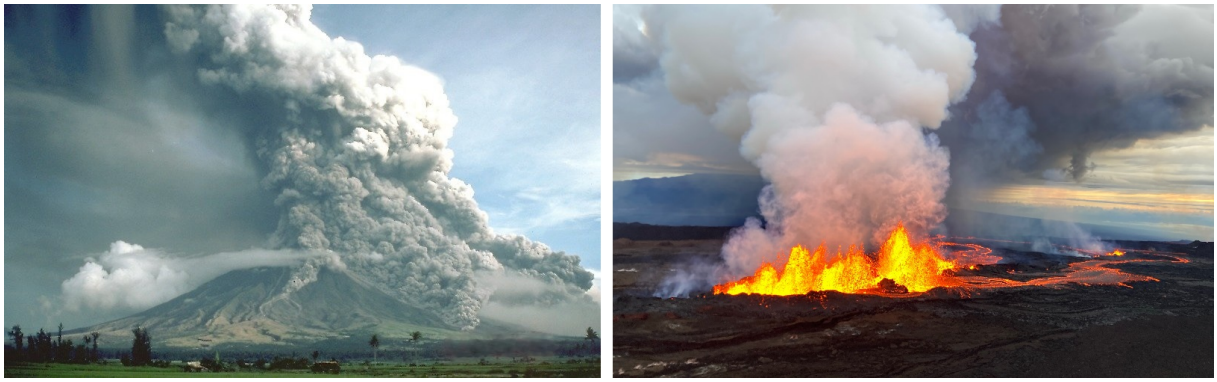

Figure 1.5: The two main types of volcanoes. On the left, a "grey" volcano, the Mayon in the Philippines, emitting a plume and pyroclastic flows. On the right, a "red" volcano, the Mauna Loa in Hawaii, emitting lava.

If fragmentation happens, it will result in explosive eruptions, ejecting gas and ashes, forming volcanic plumes and pyroclastic flows, like on 1.5-Left. These are volcanoes in a "grey" stage and are more likely to appear near subduction regions. These volcanoes are not flat like the ones in Hawaii, they form stratovolcanoes, and their shape can change a lot after a big eruption.

On the contrary, if fragmentation never happens, the volcano will mostly emit lava flows. These flows can run a long distance if the lava is very fluid, resulting in shield volcanoes like in Hawaii. We say these volcanoes are in a "red" stage, as we can see in 1.5-Right.

Note that theses two extreme cases are not the only ones: many intermediate cases can exist if the parameters of the lava vary. For instance, we can observe volcanoes with lava flows that are more viscous than in Hawaii but are still flowing. As volcanoes can appear underwater, submarine volcanoes also exist, with rapidly cooling lava that forms pillow lavas. Moreover, the type of a single eruption can change over time, and a volcano can produce different types of eruptions during its period of activity.

1.3 Methodology

For all the contributions presented in the different chapters of this thesis, I combine two complementary approaches to in order to provide a good depiction of the natural phenomena in the form of 3D animation. The first one is understanding the underlying physical mechanisms explaining and describing the phenomena I want to animate. To achieve this, I contacted a volcanologist who could explain to me these phenomena, give me advice on which simplifications I could make or not, check if my simulation algorithms sounded physically sound, and thus validate my results. I also refer to geoscience literature in my process. The second approach is a phenomenological one: rather than trying to get physically exact results, I try reproducing the phenomena as they can be observed in videos, photographs, and in real life for those who have the chance to be able to observe volcanic eruptions. Thus, I watched videos of

lava flows, volcanic plumes, and pyroclastic flows, identified specific visual characteristics to reproduce for each phenomenon, and set test cases to check if my systems can give both plausible and versatile results.

Technically speaking, I use multi-layer models in my research. In my case, implementing a multi-layer model consists of separating a natural phenomenon into several visual sub-phenomena, and attributing a layer of my model to each one of these. More precisely, a first layer would typically be a not-perfect but good enough simulation of a fluid (a lava flow, for instance) that can give a basic non-detailed animation of the phenomenon in a low computation time, ideally in real-time. The second layer would be here to add procedural visual details based on this rough simulation, still with a low computation cost (for example, details on the surface of a lava flow). There may be a retroaction applied on the first layer by the second one if specified (for instance, if the crust of a lava flow simulated in the second layer seems to tear, it may generate a new start in the flow in the first layer of simulation). In all the layers of my models, I still try to be as physically correct as possible by using physical equations or parameters observed by geoscientists in real eruptions.

1.4 Contributions

During my Ph.D., I tackled several problems in the animation of volcanic phenomena in Computer Graphics and proposed new models bringing new contributions.

From all the different volcanoes I described earlier, I only considered two types of volcanoes, trying to animate one specific aspect for each of them. I will more deeply explain how these particular types of volcanic eruptions work in the corresponding chapters of my thesis.

- First, I consider explosive eruptions, where a mixture of hot gas and ashes is ejected, and volcanic plumes and pyroclastic flows may be created depending on the initial conditions of the eruption. In Chapter 3 of this thesis, I present an efficient and consistent model that can handle both volcanic plumes and pyroclastic flows, as both can happen in the same eruption. I add procedural details to this simulation to get visual results closer to how real volcanic plumes look.

- Second, I extend this model to make it interact with an atmospheric simulation. In Chapter 4, I present a weather system I contributed to, how we make it interact with the plume system, as well as some new phenomena animated, such as cloud formation around a plume and ash rain.

- Last, I reproduce the surface effects of pahoehoe lava flows in Hawaii in Chapter 5 of my thesis, namely the formation of a crust and folds on it when it gets compressed.

Chapter 2

State of the art: the animation of natural phenomena in computer graphics

Figure 2.1: Different natural phenomena. From left to right and top to bottom: clouds, a tornado, water waves, a mountain range, an avalanche, and a sheep herd.

Natural phenomena are natural events happening in our environment, without any human intervention. They cover a wide range of different phenomena, for instance:

- astronomical phenomena, such as the movement of celestial bodies or shooting stars;
- meteorological and hydrological phenomena, like storms, wind, rain, lightning, tornadoes, as well as rainbows or northern lights, and even more complex phenomena such as the greenhouse effect;
- geological phenomena, such as earthquakes, mountain formation, erosion, volcanoes, or avalanches;

- biological phenomena, like herds behaviour, or the growth and evolution of plan ecosystems.

These phenomena are studied in many fields - mathematics, physics, biology, geology, computer science - for different purposes: modeling them, understanding them, and predicting them. Computer graphics is a very young field and takes inspiration from all the previous research done about natural phenomena in order to model them and represent them in 3D.

Several approaches exist to model and animate physical phenomena, depending on the type of phenomenon and the requirements (rapidity, precision, control...). They can be divided into three main types:

- procedural models, quick and controllable, but that have a limited range of validity. They usually model a specific phenomenon and cannot be easily generalized.

- physically-based simulations, often slower and only offering some indirect control, offer a consistent animation of phenomena. Simulation is particularly preferred when handling fluids.

- learning-based methods that are mostly deep learning based nowadays, and offer impressive results even though they require large computational capacities for training and storing training data, and are not easily controllable.

Hybrid methods are to be added to the list as many models do not fit in only one of these three categories. Indeed, many models are more complex and consist of several sub-models of different types: for instance, procedural details can be added on top of a physical simulation.

2.1 Procedural methods for shapes and motion

Procedural methods can be used for modeling and animating natural phenomena in virtual worlds, and are defined as automated methods using a certain set of rules and the associated algorithm to fullfil them. They directly model the desired result, contrary to simulation approaches that models the causes of this result and compute the effects of physical laws starting with the initial conditions. Moreover, as soon as a model is data-driven, and uses training on example data before producing results, I consider it as a learning-based system. We can point out that some models presented in previous work mix procedural, simulation and learning-based approaches: I will present such methods in Section 2.4.

2.1.1 Procedural methods for shape generation

Procedural models are particularly suited for generating new shapes automatically.

Trees: Generative models consist in a set of rules to create repetitive objects, like trees or cities. For instance, L-systems (Lin68) are particularly used for trees and plants, and result in fractal patterns (see Figure 2.2). They use production rules, which consist of iteratively replacing some parts of a geometry with other ones, often more complex. For instance, for a tree, replace a bud with a portion of a trunk and three new buds, and iterate. Other rules can be added with probabilities (giving an angle or a length to a new piece of trunk, creating more or less new buds, adding the influence of the sun...)

Figure 2.2: Trees generated with L-Systems.

Clouds: Different types of procedural methods have been used for cloud modeling, typically for the representation of stratiform clouds using slabs of multiresolution noise (BNL06; LLC+10), and convective clouds using a scaffold of implicit shapes, such as proto-particles (Ney97), warped blobs (BN04), or ellipsoids (Gar85; SSEH03; LJWcH07) that can be blended to define a volumetric density field, with an overlay of procedural noise for high-frequency detail. Such approaches have the advantage of real-time performance, due to the efficiency of noise functions and the compact support of implicit shapes. Moreover, they support user control through sketching (WBC08; SBRS10), and painting (WGM14). It is even possible to roughly shape clouds to fit a target, supplied as a mesh or photograph (DSY10; CMCM11; YLH+14). The main drawbacks of procedural approaches for modeling new shapes is the lack of realism and the difficulty to animate them consistently. However, some previous work propose procedural approaches to animate some specific natural phenomena.

2.1.2 Procedural methods for motion generation

Several procedural models for shape or motion were specifically designed with natural phenomena in mind. Let us give a few examples.

Ocean: Some models propose to animate an ocean and waves at its surface. In the work of Fournier et al. (FR86), the surface of the ocean is a parametric surface, where particles of water describe a circular

or elliptic orbit. By modifying some parameters, different types of realistic waves can thus be obtained. Hinsinger et al. (HNC02) optimized such waves representations to render an infinite ocean: they use a grid corresponding to what is seen on the screen of the user so that the closest part of the ocean gets more details and more computation time while distant water gets less details and less computation time.

Prairies: Adaptive level of details are also used to animate prairies in the work of Perbet et al. (PC01). In the region close to the camera, each blade of grass is animated separately in 3D (Figure 2.3-right). In an intermediate region, the grass is represented as semi-transparent animated billboards (Figure 2.3-middle). And from far away, a 2D texture of grass is sufficient (Figure 2.3-left).

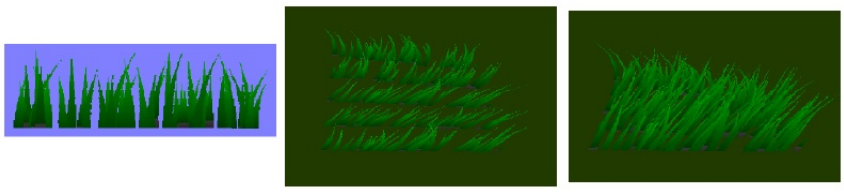

Figure 2.3: The different scales in a grass field animation (PC01).

Clouds: Procedural approaches were not only used to model clouds, but also to animate them, like in the work of Webanck et al. (WCGG18) who also add user control with keyframing interfaces. Unfortunately, realism remains an issue. With sufficient care and efforts from an artist, a convincing static cloudscape can be designed but physical plausibility often breaks down during animation. Even with specialized morphing schemes, such as optimal mass transport along anisotropic shortest paths (WCGG18) that account for input winds and terrain, the lack of physical representation makes it difficult to capture certain interactions, such as the complex behavior of clouds surrounding mountains.

2.1.3 Procedural methods for enhancing other static or animated elements

Procedural approaches are widely use to wrap textures or add details on other static or animated 3D elements.

Textures: The most common way to apply textures on 3D meshes is uv coordinates: each point of a mesh has a coordinate that corresponds to a point in a texture image linked to the mesh.

When used as textures, theses procedural models can be combined with fluid simulation using texture advection methods, as developed by Neyret (Ney03) and Guingo et al. (GLS+20). Textures can also be driven by patches moving along a velocity field on a surface, as shown by Yu et al. (YNBH11).

Texture synthesis: Another problem to tackle with textures is texture synthesis. Indeed, in many cases, using existing images causes issues: the texture may repeat itself in an obvious manner, or undesired artifacts may appear. Two good ways to synthesize textures are noise and Voronoi diagrams.

Noise. One of the most widely used techniques for generating textures representing the randomness of natural phenomena is procedural noise, and especially fractal noise, since many natural phenomena are of fractal nature. Procedural noise is a deterministic but based on a random seed function with a visible pattern and no visible periodicity. Many noises exist, but the first and most known is Perlin noise (Per85), which can be used to model clouds, terrains, fractal textures like marble or wood, etc. Some examples can be seen in Figure 2.5. I take inspiration from this kind of fractal noise to add details to volcanic plumes. Perlin noise consists of summing several instances of a semi-periodic noise at different frequencies, and smaller amplitudes the higher the frequency is, to create a fractal noise (see Figure 2.4).

$$P(x) = \sum_{k=0}^{N} \alpha^k f(\omega^k \, x) \tag{2.1}$$

In this formula, f is a pseudo-random function, N the number of octaves, α the persistence, and ω the frequency gain.

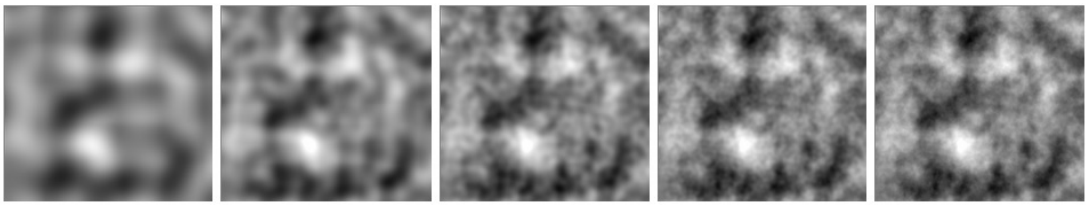

Figure 2.4: From left to right: consecutive steps of the creation of a Perlin noise, each time with the addition of a higher frequency noise.

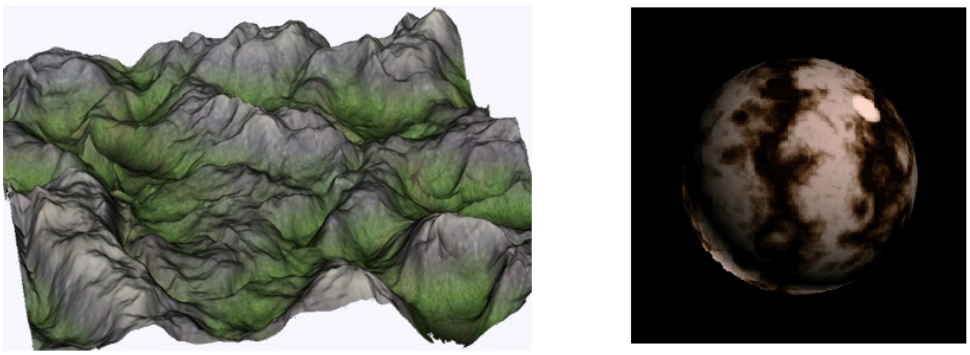

Figure 2.5: Left: terrain created using Perlin noise to compute the height of each point. Right: Volumetric marble texture created using Perlin noise.

A wide range of procedural noise functions exist and are presented in *Texturing and Modeling: A Procedural Approach* (EMP+02) or in the work of Lagae et al. (LLC+10). For instance, flow noise or curl

noise have been used to tackle the representation of fluids and smokes.

Voronoi diagrams are another method to create some particular types of textures. Given a point set named centroids, this diagram assigns the closest centroid to every point in the space: this creates a set of influence regions for every centroid (Figure 2.6-left). The border between two regions is the bisection of the segment formed by the two corresponding centroids. These diagrams can be used to represent several types of textures, such as the fur of certain animals (Figure 2.6-middle), cracks on dry soil (Figure 2.6-right), patterns on leaves, fields of crops... For example, Voronoi diagrams have been used to texture the surface of a'a lava flows by Stora et al. (SAC+99).

Figure 2.6: From left to right: a Voronoi diagram where each color region is the influence region of a centroid; a giraffe fur; a dry soil.

Other previous works achieve to procedurally generate structure in textures, like Guehl et al. (GAD+20). Texture generation will be useful for texturing lava flows, as will be described in Chapter 5 of this thesis, as a new portion of self-similar texture will gradually appear where the lava sources are.

Texturing lava flows: Procedural approaches have been used to animate lava flows, mostly for texturing, as Stora et al. (SAC+99) do. For instance, Narain et al. (NKL+07) detect the local level of detail of the surface of lava and thus used an appropriate texture for each level of detail (see Figure 2.7). Gagnon

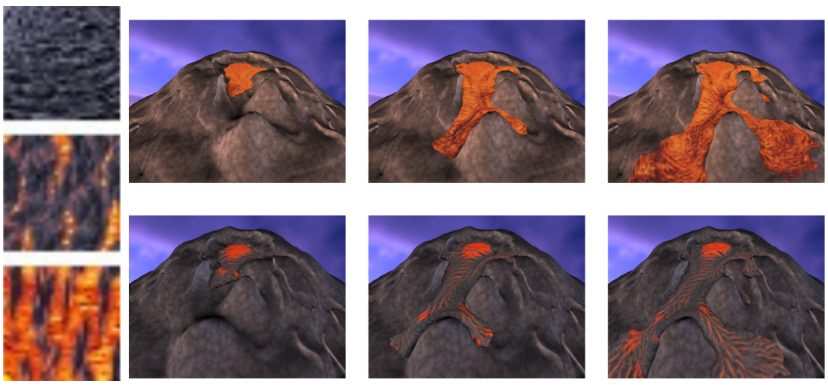

Figure 2.7: Examples of lava flow textures (NKL+07)

Figure 2.8: Texture mapping with patches and color defined by temperature on the surface of a lava flow simulation in (GGV+19)

et al. (GGV+19) use a patch-based approach for texturing a lava flow, and they add a color defined by the temperature (see Figure 2.8).

Procedural coating of a simulation: Procedural approaches cannot only be used to texture objects and simulations, but also to add geometric details on them. For instance, Rohmer et al. (RPC+10) procedurally add wrinkles on dynamic garments, and Cordonnier et al. (CCB+18) compute the formation of folds on layers of terrains to form mountain ranges.

Geometric representation of 3D objects: Finally, meshes are an explicit way to represent surfaces in 3D, and the most widely used in graphics, but implicit surfaces are another approach for the representation of 3D objects and offer advantages in some cases, for instance for the surface of fluids, clothes or organic shapes. Cani (Can93) was one of the first to use them for the animation of 3D objects, and they were developed over the years like in the work of Zanni et al. (ZBQC13). Implicit surfaces have been applied to natural phenomena to represent geological elements such as rocks or cliffs (PGP+19; PPG+20).

To conclude, procedural approaches can be used for shape or motion generation, but most importantly in my case to enhance simulations, with a wide range of application among natural phenomena.

2.2 Simulation using physically-based models

To animate many natural phenomena (water, lava, smoke...), we need in fact to animate fluids. The best way to get realistic animation of the fluids implied in natural phenomena is to reproduce the physical phenomena as closely as possible, using pre-existing physical equations. However, doing this may make simulation times way too large: lighter simulation models often have to be found, while still giving a precise enough approximation. There are two main types of simulation techniques for fluids: Eulerian and Lagrangian methods. The former use a simulation grid where we follow the evolution of physical

quantities in each cell of the grid; whereas the latter follows physical particles evolving freely in the space with attached physical quantities.

2.2.1 Eulerian methods

The most obvious way to do an Eulerian simulation is to use brute force without any optimization and apply the Navier-Stokes equation

$$\begin{gathered} \frac{\partial u}{\partial t} = -\frac{1}{\rho}\nabla \mathrm{p} + f - (u \cdot \nabla)u + \nu \triangle u \\ div(u) = 0 \end{gathered} \tag{2.2}$$

on the grid with the finite elements method, for instance, in Foster et al. (FM97) for the simulation of a gas. Here, u is the velocity, p the pressure, ρ the density, f the forces and ν the viscosity. However, other algorithms are faster, more stable, and give good or better results.

The Stable Fluids method's main idea, as it was stated first by Stam (Sta99), is to remove the pressure term of the Navier-Stokes equation $\frac{1}{\rho}\nabla \mathrm{p}$, and replace it by an explicit projection on a divergence free vector field. Also, the different parts of the equation are treated separately to compute the new velocity in each cell: first, add the forces f, then do the diffusion $\nu \triangle u$, then project on a divergence free vector field, and finally advect the velocity $(u \cdot \nabla)u$. Stable Fluids has widely been used since then, as it is an efficient and robust method.

Algorithm 1 Stable Fluids: a simulation step

1: Add forces
2: Solve diffusion
3: Project on a divergence free vector field
4: Advect velocity along itself

When it comes to animating natural phenomena, the stable fluid method may not be enough. Indeed, we may need to be able to handle highly viscous fluids, fluids that can solidify or solids that can melt, to simulate some materials like sand or lava. Carlson et al. (CMVHT02) proposed one of the first Eulerian methods to simulate this type of fluid, and Takahashi et al. (TL19) proposed an approach to simulate honey, which is a very viscous fluid.

Eulerian methods have been used to simulate all kinds of natural phenomena:

1. **Erosion and lava:** Mei et al. (MDH07) used a grid to model erosion. In each cell, there is a certain height of terrain and water. Knowing the height of both one cell and its neighbours, using a shallow waters algorithm the displacement of the water can be computed. The water transports a small amount of the terrain while going from one cell to another, which mimics how erosion

works. This method was also used for animating lava flows by Chahi and Sahy in their Volcano Simulator for the volcano museum La Cité du Volcan in La Réunion (CS15).

2. **Smoke:** Smoke simulations were extensively studied in CG, mostly using Eulerian grid-based methods (KVH84; Sta99; FSJ01). Until recently, an impressive body of works has been dedicated to the improvement of these methods, in either accuracy, robustness, or efficiency (see (TKP13) for a comprehensive survey). Indeed, smoke simulation at large spatial scales can be highly intensive in memory and computation usage. Performances were improved using moving grids able to follow the flow motion (SCLP04), or sub-grids able to merge and split (PCCP05).

3. **Weather phenomena:** Closer to the high-speed, turbulent, and dense gas ejected by volcanoes, weather phenomena such as cloud formation share the challenge of combining large spatial extents with small-scale details. Similar to cloud simulation methods (KVH84; HMP+20; HHP+21), we model thermal transfer and buoyancy. However, existing solutions cannot be reused for animating volcanic plumes. Indeed, volcanic eruptions involve much higher velocities, drastic density and buoyancy variations, and high turbulence, which may either cause stability or integrity issues or would require too small time steps to achieve interactive simulation.

4. **Volcanic plumes:** The only work that tackled the simulation of volcanic columns was the early work from Mizuno et al. (MDN02). An Eulerian simulation on a coarse cubic grid of 100^3 voxels was used to compute the dynamics of ejected material. A procedural decrease, function of the height, was used for the density of the material. It did not take into account the velocity of the ejection, thus failing to capture the interaction between high-speed ejected material and the surrounding air. In addition, the use of a fixed cubic grid limited the representation of small-scale details while being computationally intensive. In contrast, our approach in Chapter 3 relies on ejection-air interaction laws. It is therefore able to automatically capture both buoyant ejection and pyroclastic flows. Moreover, it enables a real-time visual simulation of volcanic ejections, while spanning from small to large spatial scales.

2.2.2 Lagrangian methods

Lennard-Jones potentials were used for the first Lagrangian approach in Computer Graphics (TPF89; Ton91). They describe the interaction between two particles, and whether they are attracting or repulsing each other depending on the distance between them. This method has been used to simulate viscous liquids, soft solids or melting materials. Lennard-Jones forces can be used to animate the motion of birds or herds with the Boids method (Rey87). In this model, the behavior of each individual

is simplified with only three rules: separation (if two animals are too close, they move away from each other), alignment (if two individuals a reasonably close, they go in the same direction) and cohesion (if two individuals are not close enough and not too far away from each other, they tend to get closer).

The Smoothed Particle Hydrodynamics (SPH) method has first been introduced in astrophysics by (Luc77) and (GM77). Unlike Lennard-Jones potentials where forces had to be set up by hand, the forces in SPH derive from pressure (for instance), which makes it possible to sample the same fluid at different scales. It was applied in Computer Graphics by Stam et al. (SF95), and then by Desbruns et al. (DG95; DC96) to simulate highly deformable materials. It is a purely Lagrangian technique, where the following formula can compute any physical quantity A anywhere in the space, knowing its value where the simulated particles are:

$$A(p) = \sum_i \frac{m_i}{\rho_i} A(p_i) W_h(p - p_i) \tag{2.3}$$

W_h is a kernel weighting the influence of the neighbouring particles, knowing their distance from the point where A is computed. For instance, it can have the following shape, where h is the influence radius of the particles:

$$W_h(p) = \frac{315}{64\,\pi\,h^3} \left(1 - \left(\frac{\|p\|}{h}\right)^2\right)^3, \quad \|p\| \leq h \tag{2.4}$$

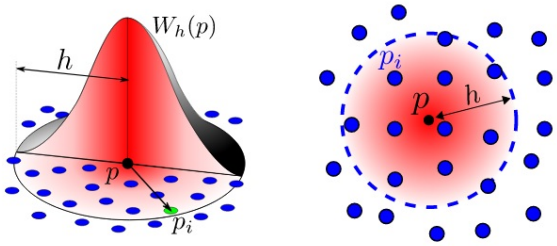

Figure 2.9: The kernel used in the SPH simulation. The further a particle is, the less it influences the density.

This formula is then used to compute the density ρ where each particle i is:

$$\rho(p_i) = \sum_j m_j W_h(p_i - p_j) \tag{2.5}$$

This density is converted into a pressure using the Tait equation: $P(p) = K(\frac{\rho(p)}{\rho_0} - 1)^\alpha, \rho > \rho_0$, where ρ_0 is the rest density and K a stiffness constant. Once the pressure of the fluid is known at every particle, we can compute their acceleration using the Navier-Stokes equation. There are three parts in this equation:

one due to gravity, one due to the pressure force, and one due to the viscosity ν:

$$v_i'(t) = g - \sum_j m_j \left(\frac{\mathrm{p}_i}{\rho_i^2} + \frac{\mathrm{p}_j}{\rho_j^2} \right) \nabla W_h(p_i - p_j) + 2\nu \sum_j \frac{m_j}{\rho_j} \frac{(p_i - p_j) \cdot (v_i - v_j)}{\|p_i - p_j\|^2 + \epsilon h^2} \nabla W_h(p_i - p_j) \tag{2.6}$$

Last, an integration step gives the new velocity and position of each particle.

Algorithm 2 SPH: a simulation step

1: Compute density
2: Compute pressure
3: Compute acceleration
4: Integration to obtain the velocity and position

Clavet et al. (CBP05) proposed an extension of the SPH method, improving its flexibility and robustness, by introducing a double kernel relaxation and interactions between particles using springs. At each simulation step, each particle is moved along its velocity, while taking gravity the viscosity into account. This new position is then edited by successively applying to the particle the influence of the springs, the double-density relaxation, and the collisions.

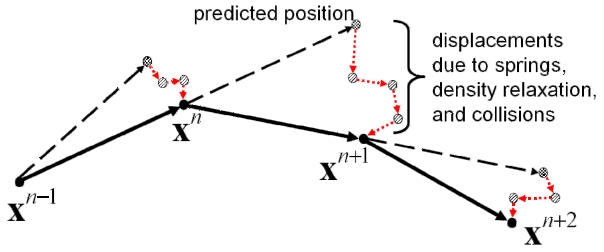

Figure 2.10: Computation of the next position for a particles in (CBP05).

The springs between the particles introduce elasticity and plasticity in the fluid. Springs are created between neighbouring particles; their rest length is updated if these particles get closer or farther; and they are deleted if the particles get too far away. The double density relaxation takes two different measures of the density into account: the classic density and a close density that prevents the particles from forming clusters by giving a higher weight to closer particles. Therefore, the particles that are too close are repulsed away.

Adaptive particles is another extension of the SPH method introduced by Desbruns et al. (DC99), where the size of particles can change dynamically: either decrease in complex areas where more details are needed (surface of a fluid, turbulence...), increasing the number of particles, or increase where the fluid is more homogeneous, decreasing the number of particles. This is a more efficient way to have a more precise fluid simulation. This way, a particle in an area where density varies a lot will subdivide into several particles while preserving the mass, volume, and velocity of the fluid. On the contrary, if

Algorithm 3 SPH extension in (CBP05): one simulation step

```
for all particles do
    Apply gravity to the velocity
end for
Apply viscosity to the velocity of the particles
for all particles do
    Save the previous position
    Move to the new position predicted by the velocity
end for
Add/remove springs, edit their rest length if needed
Modify positions due to the springs
Modify positions due to the double-density relaxation
Modify positions due to collisions
for all particles do
    Compute the new velocity
end for
```

many small particles are situated in an area where the density is homogeneous, they will fuse into a bigger particle (see Figure 2.11).

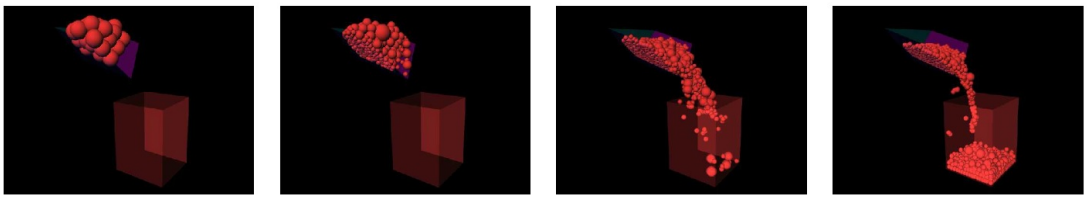

Figure 2.11: Adaptive particles in (DC99).

This method was improved by several other contributions. Keiser et al. (KAG+06) used only a discrete set of particle sizes and added virtual transitory particles at the interfaces between different-sized particles, to make the computation easier and faster. Adams et al. (APKG07) define areas in the fluid where the resolution has to be smaller or bigger depending on the distance from the surface of the fluid and the level of details asked: this way, they make sure that there will be more particles on the surface and less in the depth. Solenthaler et al. (SG11) propose a two-resolution fluid, consisting of two distinct coupled simulations of a low-resolution fluid and a high-resolution one. Orthmann et al. (OK12) introduce a temporal blending in the subdivision and fusion of particles to ensure continuity. Winchenbach et al. (WHK17) also define an ideal particle size everywhere in the fluid like Adams et al., but this size evolves continuously and is not restricted to a limited set of values. If a particle's size is close to its ideal size, it does not change; if it is slightly smaller or bigger, the mass is balanced with neighbouring particles; if it is too small, its mass is entirely distributed between close particles and the particle disappears; finally, if it is too big, a subdivision happens.

Akinci et al. (AIA+12) describe a two-way solid/rigid coupling for incompressible SPH, by sampling the surface of rigid objects with boundary particles interacting with the fluid. SPH has also been adapted

for highly viscous fluids, like in Weiler et al. (WKBB18).

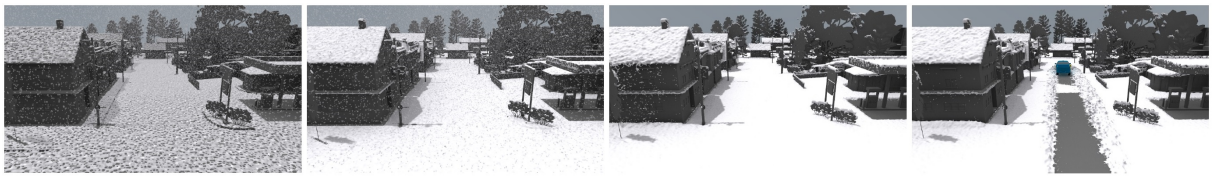

Figure 2.12: Snow simulation with SPH in in (GHB+20).

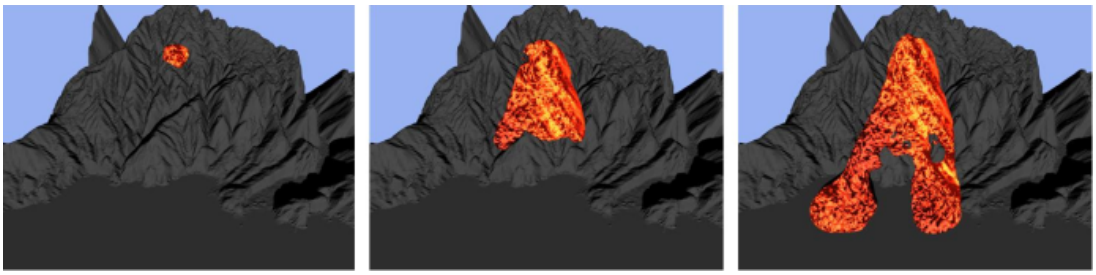

Figure 2.13: Lava simulation in (CBL+09).

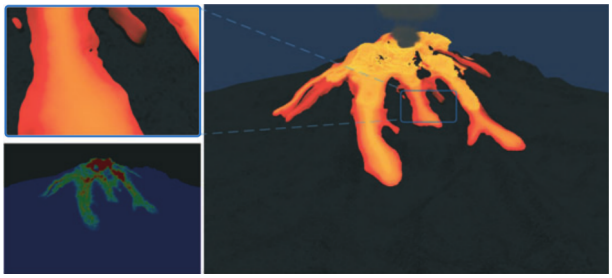

Figure 2.14: Lava flow simulation in (ZKL+17), coupled with a light smoke.

Here are some example of applications of Lagrangian fluids for natural phenomena:

1. **Snow:** Gissler et al. (GHB+20) propose an application of SPH for snow simulation by doing an implicit compressible simulation (see Figure 2.12). In this work, an implicit compressible pressure solver handles the snow compression. The computation of the acceleration due to the shear stress is also done with an implicit formulation.

2. **Smoke and clouds:** Lagrangian approaches have also been used to model smoke or clouds. Indeed, they are better suited to open domains than Eulerian grids but prove costly as well when many particles need to be used. Several methods relied on the enhancement of coarse-grain particle systems with procedural models, aimed at conveying details. An early example (Ney97) represents air parcels using bubble-like particles. This approach was also used in VFX: PDI and Digital Domain used rolling sphere representations to model dragon flames and avalanches (Dre01; Kap03). More recently, Welton et al. (WBDY15) and Barbosa et al. (FBDY15) incorporate dynamic

splitting and merging strategies to cope with a variable density over time. In addition to their suitability for large, open domains, Lagrangian methods allow for the accurate modeling of vorticity in billowing phenomena: Vortex particles (CK00) enable to accurately locate vortices, concentrating the simulation cost to useful regions only. Angelidis (ANSN06) introduced vortex filaments instead of particles, to improve the conservation of vorticity energy, momentum, and strength. Despite performance improvements (WP10), such simulation would however get costly for large and detailed billowing objects such ad volcanic columns.

3. **Lava and volcanoes:** Stora et al. (SAC+99) were the first to use SPH to simulate lava flows, with a layered model I will describe more precisely in Section 2.4. Chang et al. (CBL+09) also used SPH to simulate lava flows as viscoelastic fluids, as seen in Figure 2.13. They modeled heat diffusion, thus allowing their fluids to melt or harden, as it is the case for lava depending on its temperature. However, this was not sufficient to create a clear crust and folding as I do in my work. Zhang et al. (ZKL+17) animate a volcano with SPH. More precisely, they used predictive-corrective incompressible SPH to enforce incompressibility for the lava simulation. They did not only focus on the lava flow animation but added particles for the ground for taking ground friction into account, as well as a smoke emission, also as Lagrangian particles. What's more, they allowed phase transition for the lava and distinguished solid and liquid lava particles, introducing fluid-rigid coupling between these two categories and with the ground (see Figure 2.14).

2.2.3 Hybrid methods

As neither Eulerian and Lagrangian methods are perfect and both have flaws (for instance, the handling of boundaries for both kinds of methods), hybrid methods have been more and more developed.

One of the first method to do this is the Level Set algorithm by Foster et al. (FF01). The idea is to do an Eulerian simulation and add a surface boundary with implicit surfaces that will be tracked during the simulation. More precisely, a volume $\Omega = \{p \in \mathbb{R}^3, \varphi(p) = 0\}$ is defined, where φ is a function of the 3D space. At each time step, the Navier-Stokes equation is solved within this volume; and then, Ω is deformed using the implicit equation $\frac{\partial \varphi}{\partial t} + u \cdot \nabla \varphi = 0$.

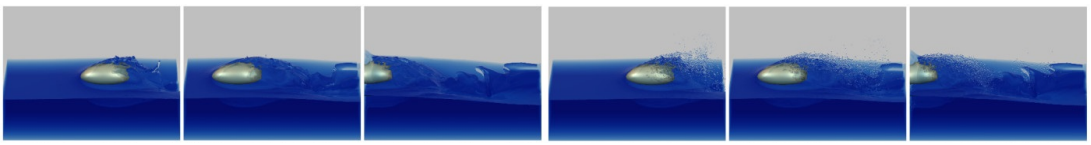

Figure 2.15: Rendering of Level Set method in (FF01).

However, this method causes smoothing and a loss of volume. Thus, Enright et al. (EFFM02; EMF02)

further develop a Particle Level Set method where Lagrangian marker particles rebuild the surface in low-resolution regions.

PIC (Particle-In-Cell) and FLIP (Fluid-Implicit-Particle) are two hybrid methods using both a grid and particles at the same time, mixing Eulerian and Lagrangian approaches, as particles work better for advection while a grid is best for every other step (forces, pressure, viscosity).

They both use a Marker and Cell (MAC) grid, where scalar values such as pressure or density are stored in the middle of the cells, and velocity components on the edges, which improves accuracy and stability (see Figure 2.16-left).

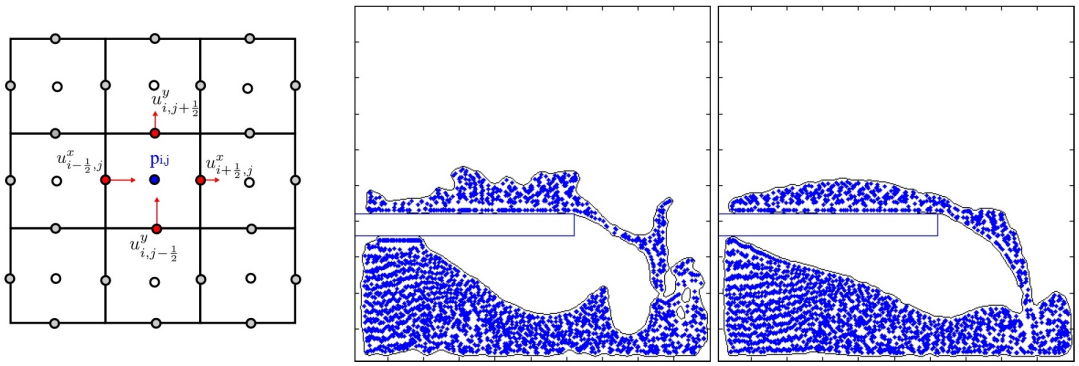

Figure 2.16: Left: The MAC grid. Middle: PIC simulation. Right: FLIP simulation.

The PIC method, introduced by Harlow (Har63), simply transfers the velocity from the grid to the particles. PIC simulations are quite stable and smoothed out (see Figure 2.16-middle).

Algorithm 4 PIC: a simulation step

1: At each MAC grid node, compute a weighted average of the nearby particle velocities
2: Do the standard Eulerian steps on the grid (pressure, forces, viscosity) except advection
3: Interpolate the new grid velocities to transfer them to the particles
4: Advect the particles

The FLIP method, first developed by Harlow (HW65), adds the velocity difference in the grid from the grid to the particles. FLIP simulations have more details than PIC ones and few dissipation (see Figure 2.16-right).

Algorithm 5 FLIP: a simulation step

1: At each MAC grid node, compute a weighted average of the nearby particle velocities
2: Save the grid velocities
3: Do the standard Eulerian steps on the grid (pressure, forces, viscosity) except advection
4: Subtract the new grid velocities from the saved ones and add the interpolated difference to the particles
5: Advect the particles

Finally, the Material Point Method (MPM) is an extension to PIC/FLIP allowing a wider variety

of materials with different properties. Indeed, it can be used for solids, liquids, or gas, as well as for simulating multi-phase interactions. It has been more and more used in Computer Graphics during the last decade.

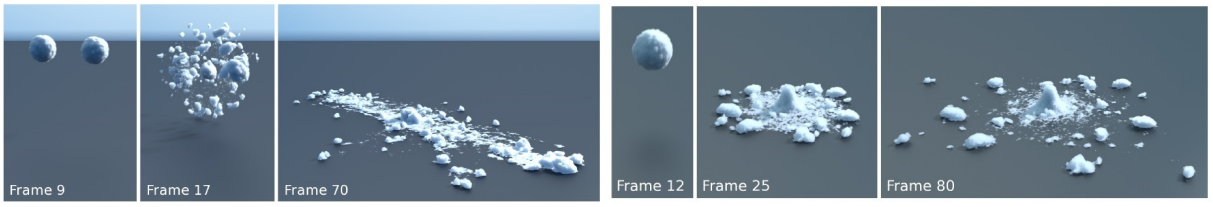

Figure 2.17: Snow simulation with MPM in (SSC+13) ©Disney.

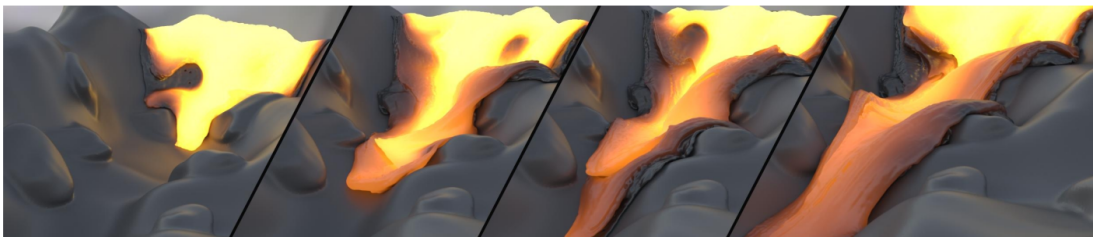

Figure 2.18: A pahoehoe lava flow animation using MPM in (SSJ+14) ©Disney.

Like Eulerian and Lagrangian approaches, the different hybrid methods have been used to animate natural phenomena:

1. **Sand:** PIC was used by Zhu et al. in (ZB05) to simulate sand.

2. **Ocean:** FLIP was used, for example by Huang et al. (HQT+21) to animate an ocean.

3. **Snow:** Using MPM, Stomakhin et al. from Walt Disney Animation Studios (SSC+13) animated snow, which was used in the movie *Frozen*. Thanks to MPM, they were able to model the compressibility, plasticity, and hardening of snow, and they could show snowballs fracturing, being sticky (see Figure 2.17), as well as interactions between snow and characters walking in it.

4. **Multi-phase materials:** Stomakhin et al. have then used MPM again to simulate melting, freezing, or solidifying materials like butter, chocolate, or lava flows (SSJ+14). They simulate phase changes of these materials using a heat solver, altering their mechanical parameters knowing the temperature. They obtained nice animation of pahoehoe lava flows in Figure 2.18. However, even if the physics of the flow are impressive, they miss the surface effects such as crust formation and wrinkles needed to give a realistic view of such a flow. This may be hard to obtain with a single physical simulation, so in my work I rather used a less powerful simulation but with procedural details on another layer.

In conclusion, physically-based simulation is necessary in my case for the animation of natural phenomena implying fluids such as lava or volcanic plumes. As I need my simulation to be fast, I can use lightweight Eulerian, Lagrangian or hybrid approaches. But simulation on its own may not be sufficient, and I need to combine it with other types of approaches.

2.3 Learning-based approaches

Learning-based approaches have been applied in natural phenomena too: older models are example-based, like the work of Kwatra et al. (KAK+06), where the explore the advection of textures on fluids using example textures as inputs.

Since deep learning became more and more popular in the last few years, the use of learning-based approaches for natural phenomena modeling has increased, especially for fluid animation. Indeed, since such simulations cost much computation time and space for every run, using neural networks to learn on real phenomena makes all the resources go into the training, while producing new animations afterwards becomes lighter.

For instance, Xiao et al. (XZWY20) used convolutional neural networks (CNN) to train a Poisson solver for Eulerian fluid simulations. They got as a result much faster simulations, but the Poisson solver was less precise than the other solvers they compared with, and they advised not to use their solver if strict numerical accuracy is needed. Learning-based methods can also be used for detail enhancement, like in (WXCT19) where they used a GAN to get better small-scale details (see Figure 2.19).

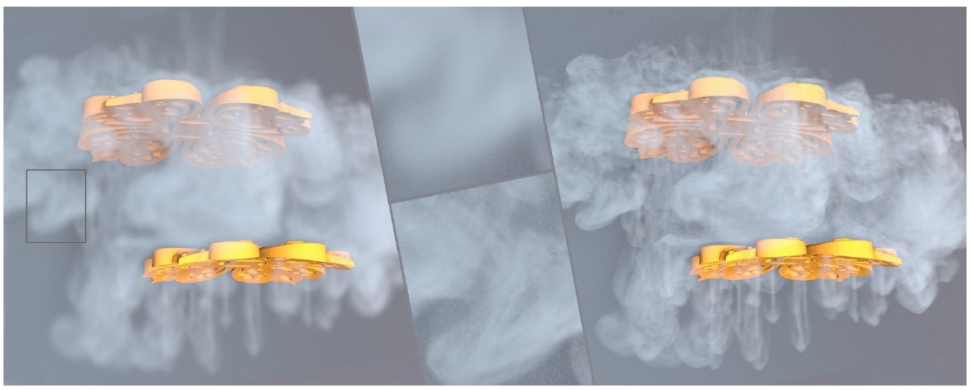

Figure 2.19: Super Resolution on a fluid flow obtain with a GAN in (WXCT19).

Many other methods cover learning-based fluid animation techniques: Chen et al.(CWWY21) survey a wide range of data-driven techniques in this field, covering Eulerian, Lagrangian, and hybrid simulations as well as many applications like detail enhancement, animation synthesis or fluid control.

Tackling another field of application, Kim et al. (KHW+22) used a CNN to model smoke based on artist sketches. They achieve density reconstruction of very non-physical shapes for smoke, which

would be complicated to do with procedural or physically-based approaches.

Finally, I did not cover rendering techniques much in this chapter, but it has to be noticed that machine learning can bring a lot of realism in the animation of natural phenomena, especially when it comes to certain hard-to-render objects such as clouds. Indeed, rendering can also be long to achieve with classical approaches, particularly for clouds or snow where the path of light can be quite complex, and learning-based models can be better for this purpose. For instance, deep neural networks have seen use as an alternative solution for cloud rendering (KMM+17).

In my work, I chose not to use learning-based approaches as I preferred using methods that did not require training, considering the lack of simulation data of volcanic eruptions. Moreover, they were not as suited as layered models combining procedural and physical sub-models in my case.

2.4 Layered approaches

Very often, one set of methods is not enough to model and animate natural phenomena. Indeed they may be too complex, as they show several scales of details of different nature. In this case, they can often be decomposed into sub-phenomena, and the best way to tackle these is not necessarily the same. In this section, I will give some examples of different hybrid approaches, leading to the set up of multi-models that combine several of the strategies we already listed.

Layered models are commonly used when a complete natural phenomenon is to be represented, and not only a sub-phenomena. The most common way to achieve it is to combine a physically-based simulation and procedurally generated details. Here are some examples:

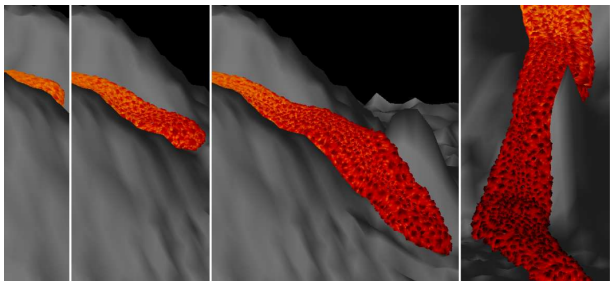

Figure 2.20: A'a lava flow in (SAC+99).

1. **Deformable objects and soft substances** can be simulated (Can93; DG95) with a Lagrangian simulation and deformable implicit surfaces communicating forces to particles if collisions occur.

2. **Lava flows:** Stora et al. (SAC+99) propose to specifically animate a'a lava flows by capturing several sub-phenomena in it. First, for the fluid simulation in itself, they use SPH and heat transfer between the lava particles, and with the environment. Then, to mesh it, they use implicit surfaces

and projected sample points - one for each particle on the flow - on the isosurface. To obtain temporal coherence, they use a Voronoi diagram made from these sample points. Finally, they generate a texture by computing displacement and color on the mesh with a pseudo-periodic noise inspired by Perlin noise. A result is shown in Figure 2.20.

3. **Wrinkles on a garment** can also be modeled and animated using simulation and procedural methods in Rohmer et al. (RPC+10): the size and shape of the wrinkles are defined using the stretch tensor of the fabric and implicit surfaces. I use a similar approach to represent the crust at the surface of lava flows.

4. **Terrain generation** can be done with simulation and procedural models, like in Cordonnier at al. (CBC+16) where tectonic uplift and fluvial erosion are modeled to compute the height of the terrain after several iterations.

5. **Waves:** Brousset et al. (BDM+16) couple a procedural model with simulation to animate waves. Indeed, they set up a new procedural wave model to animate breaking waves more specifically, improving other wave models presented in the procedural methods section. They deduce forces from this, and they simulate an SPH fluid constrained by these forces. With this hybrid model, they are able to animate specific effects such as breaking waves, or curved wavefronts.

6. **Clouds:** In cloud animation, hybrid schemes overcome the realism issues of procedural methods and the scale issues of simulation, by coupling coarse-scale simulation with detail amplification using volumetric noise. For instance, Goswami et al. (GN17) use Lagrangian simulation of air parcels with noise-based procedural detail for convective cloud simulation.

Learning and procedural models may also be coupled. Indeed, learning-based methods may not be self-sufficient and require procedural refinement. This is the case in the work of Guérin et al. (GDG+17), where terrain modeling authoring is achieved using GANs. However, the output lacked smaller scale details and had to be amplified using a patch-based procedural method from (GDGP16) (see Figure 2.21).

For animating natural phenomena such as lava flows and volcanic plumes, layered models combining physically-based simulation and procedural details seem the best suited. Indeed, a simple simulation is enough to capture the general physical behaviour of the phenomena, and a procedural layer can reproduce details for a specific phenomenon, as I described in this section.

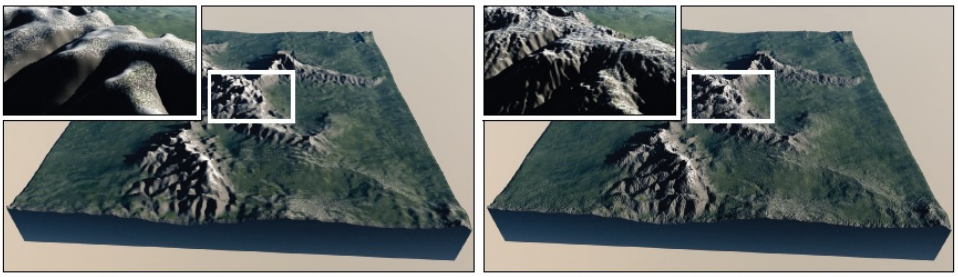

Figure 2.21: Left: a terrain obtained with a GAN in (GDG+17) before amplification. Right: the same terrain after amplification.

2.5 Conclusion

Even if I can take inspiration from procedural and physically-based approaches in my work, they are not sufficient or too costly for my goals. Moreover, I choose not to use learning-based techniques as they require a lot of computation resources.

In this thesis, as I want to model and animate consistent and complete volcanic phenomena, I will therefore use layered models as I described in the last section. Indeed, they seem the most well-suited for me. In all of the work I present in the remainder of this thesis, I use layered models: the first layer is a coarse simulation, that can give the general trajectory of the fluids I use, whether they are gas or lava flows. These physical simulations do not need to give small-scale details; indeed, I use a second and sometimes a third layer to refine and amplify my animations with procedural methods specifically made for reproducing very precise phenomena, such as the visual appearance of a volcanic plume or the folds appearing at the surface of a pahoehoe lava flow.

Chapter 3

Interactive simulation of plume and pyroclastic volcanic ejections

3.1 Introduction

As many myths and legends attest, volcanic eruptions are one of the natural phenomena that struck humans the most, due to their unique combination of stunning beauty and destructive power. As still popular storytelling element in movies and games, volcanic phenomena are nowadays monitored and analyzed for natural risks prevention, accurately modeled for better understanding and prediction, as well as explained to children at school or to the public in museums. Several of these applications, from education, museography, and communication about risks, to special effects (VFX) previsualization and games, share the need for an efficient, yet visually realistic, animated model for volcano eruption. In this chapter, we tackle a part of this challenge, namely the interactive simulation of volcano's ejection columns and pyroclastic flows.

While volcano ejection was not much studied in Computer Graphics (CG) so far, smoke on one hand,

Figure 3.1: Interactive simulation (left) and off-line rendering of ejection phenomena during volcanic eruption: Rising column under the wind (middle); Pyroclastic flow (right), where secondary columns are generated along the way.

and convective clouds on the other hand attracted a lot of work, leading to efficient simulation methods for combustion and condensation phenomena, respectively, using Eulerian methods in bounded domains. Existing software such as SideFX-Houdini (Sid21) propose to reuse these as a default model for volcanic ejection. In addition to not allowing real-time interaction, these simulations however fail to capture the specific dynamics and visual appearance of real eruption recordings.

Indeed, to be plausible, simulations of volcanic ejection need to handle the two following specific features: First, the combination of extremely large size scales – as volcanic columns can expand for tens of kilometers vertically and hundreds horizontally – with fine details at the scale of a meter starting from ground level. Catching these two key scales is mandatory to allow close-ups and fly-overs. Second, the specific dynamic behavior of plume columns, as the eruptive material is initially expelled at high speed – close to the speed of sound – and with a density hundred times higher than the density of the surrounding atmosphere. These extreme initial conditions lead to two unique emerging behaviors, drastically different from those of clouds or smoke: either a pyroclastic flow spreading down the slopes of the volcano, or an ascending plume propagating upward and finally spreading side-way at high altitude. To capture these specific large-scale behaviors, the reciprocal interaction between the highly turbulent, dense flow and the surrounding atmosphere must be accurately computed.

Efficiently capturing such inherent relationships between small and large scales, within an open environment, would be extremely difficult using traditional grid-based simulations. Therefore, we resort to the use of a layered animated model, which can be seen as the combination of simple and consistently coupled sub-models. More precisely, we combine minimalist Lagrangian simulation at medium scale, used to represent dynamic horizontal slices of material ejected by the volcano and interacting with the surrounding air, with a procedural model that enhances the visual animation of the turbulent flow with multi-resolution details.

Our technical contributions are threefold:

- We propose an efficient, yet consistent model for volcanic column dynamics at medium and large scales, based on a simple Lagrangian simulation inspired from prior knowledge and behavioral laws from volcanology studies.
- We introduce a procedural enrichment inspired by Kolmogorov cascades, to capture the complex motion of turbulent ashes and represent fine details up to a meter.
- We extend the previous two-layers model to capture pyroclastic flow propagation and the resulting sub-columns.

As our results show (see Figure 3.1), our model is able to convey the essence of volcanic ejections dynamics, from ascending columns interacting with the wind to pyroclastic flows, while achieving visual

realism thanks to the animated procedural details.

We validate the consistency of our solution through comparison with predictive models from volcanology literature, regarding the evolution of velocity and density over time and height, the shape profile of columns, and the maximal altitude of the plume. Lastly, we discuss two rendering techniques: a real-time pre-visualization method well suited to artistic modeling pipelines where manual tuning should be facilitated to enable fine-grain control; and off-line, volumetric rendering, enabling to compute realistic animations that can be visually compared with real images.

The content of this chapter led to a publication at the ACM Symposium on Interactive 3D Graphics and Games (I3D) (LRC+22), as well as a presentation at the French Workshop on Animation and Simulation (GTAS) in 2021.

[Maud Lastic, Damien Rohmer, Guillaume Cordonnier, Claude Jaupart, Fabrice Neyret and Marie-Paule Cani. Interactive simulation of plume and pyroclastic volcanic ejections. Proceedings of the ACM on Computer Graphics and Interactive Techniques, 5(1):1-15, May 2022.]

3.2 Related work

We have already exposed previous work for this chapter in Chapter 2, but here is a quick reminder of the main paper that covers volcanic ejections or close phenomena. To our knowledge, the Eulerian model from Mizuno et al. (MDN02) is the only work in Computer Graphics to simulate a volcanic plume. However, they did not take some specific phenomena implied in volcanic ejections physics, such as air entrainment. Indeed, it causes the raises of the plume, it increases its radius, and it decreases its density, which are important visual characteristics of volcanic plumes.

Apart from that, cloud animation - and more specifically when it covers cumulonimbus formation - is a close domain of research. Indeed, like volcanic plumes, cumulonimbus rise high in the atmosphere due to the effect of convection. For instance, (HMP+20; HHP+21; WCGG18) reproduce these types of clouds, where important convection happens. However, the temperatures and velocities involved in volcanic plumes are much higher, which requires more precision and smaller time steps.

3.2.1 Volcanic column models in geoscience

The formation of volcanic plumes was extensively studied in natural sciences. This literature has not been much exploited in computer graphics until now, but it is necessary to understand the underlying physical phenomena, and we can take inspiration from certain models described in the geoscience literature.

Volcanic columns are modeled as multiphasic flows, simulated using CFD codes on discretized domains (MFi21; DR06). These models are extremely costly and require hours to days of computations. As volcanic plumes can rise over tens of kilometers before reaching their equilibrium altitude, coarse spatial resolutions are used, with voxels typically spanning a hundred meters. Therefore, only large-scale phenomena such as the plume expansion as well as macroscopic chemical and thermodynamic exchanges are captured (TGL+05; SCC+16), while smaller scale phenomena are only averaged. Spatial resolution is too coarse to reproduce the visual appearance observed on pictures of real volcanic ejections, where small vortices of about a meter wide can be distinguished. These solutions are therefore not well adapted to CG applications such as VFX, where visual details, efficient computation, and controllable results are of utmost importance.

Nevertheless, these simulations are based on physical laws expressing the macroscopic interaction between the volcanic column and the ambient air (MTT56; SK10), or the effect of the wind (SK15). We reuse these laws, which are of great interest for real-time visual simulation. In addition, general knowledge on volcanic column shape and dynamics is typically reported by volcanologists as abacuses and measured curves (Spa86; Woo88; Jau96; WHPS13; CB15). These curves depict global phenomena such as the general profile of a plume or the temporal evolution of the ascending velocity. Being generally specified for a specific set of standard conditions, they are not sufficient to generate a full, 3D geometric representation. In this work, we use them as references for validating the physical plausibility of our results.

Note that we developed our solution in collaboration with a geo-scientist. The fact that we use the same physical laws can make our interactive model relevant in their field, for instance to quickly test various parameters values before launching a more precise simulation. Moreover, in contrast with coarse Eulerian-based simulations which fail to capture interactions with the actual terrain topography, we make use of an accurate terrain model, which can be a plus in predictive applications.

3.3 Overview

Our algorithm is based on knowledge in the volcanology of eruptive columns, which I first explain, before describing our model.

3.3.1 Explosive eruption columns in nature

Volcanic gas and ash particles are released from volcanic vents at high velocity (typically 50 to 200 m/s) as a high density mixture (a few hundred kg/m^3). During an initial *jet phase*, the volcanic mixture is principally propelled from its initial momentum. As the mixture is denser than the atmosphere, it

slows down rapidly due to its weight. However, the mixture flows in a turbulent regime, and therefore constantly *entrains* the surrounding air, which gets heated in contact with hot ash particles. Therefore, the fraction of gas increases, which results in a decrease of the mixture density, and thus in an expansion of the column volume. Two possible regimes, presented next, can therefore emerge.

If the rising mixture becomes less dense than the surrounding cooler air, it becomes propelled by buoyancy. The initial jet column then turns into a characteristic volcanic plume mainly driven by buoyancy. This eruption type is called *Plinian regime*. As the vertical decrease of density is slower for the rising parcels than for the surrounding atmosphere (due to the effect of humidity in the troposphere, and then of the heating due to UVs absorption by the ozone layer in the stratosphere), this rise can continue until reaching a maximal altitude characterized by a neutral buoyancy level –i.e. density equality– which can typically happen at a height of tens of kilometers. Near this maximal altitude, the mixture spreads laterally and typically expands along atmospheric winds.

In reverse, if the upward velocity vanishes while the mixture is still denser than the surrounding air, it falls back and generates a pyroclastic flow, also called the *Peléan regime* of the eruption. As the mixture slides and spreads along the terrain slope, the heaviest particles and some ashes settle onto the ground. Therefore, the density of the pyroclastic flow decreases, eventually below the density of the surrounding air. In this case, the buoyancy force allows the mixture to rise again, which leads to the appearance of sub-columns along the spreading region. These sub-columns, however, have a smaller radius and are slower than the one directly above the vent because their initial velocity and turbulence are lower.

Note that a volcano can produce these two ejection regimes and that the type of ejection may even change over time. A famous example is the eruption of Mount Vesuvius, 79 AD, which started as a *Plinian regime* before switching suddenly to *Peléan* leading to the destruction of Pompeii by the pyroclastic flow.

3.3.2 A layered model for volcanic ejection

The main insight of our model is to decouple the computation of the global dynamic behavior of the mixture ejected by a volcano from the generation of local motion within the resulting, turbulent flow. Indeed, global dynamics can be efficiently evaluated using a geometrically simplified, yet physically-accurate model. In reverse, a procedural model, carefully coupled with the first one for consistency, is sufficient to generate the necessary amount of animated details at the benefit of low computational costs. These two coupled sub-models form a layered representation of the whole phenomenon, detailed below.

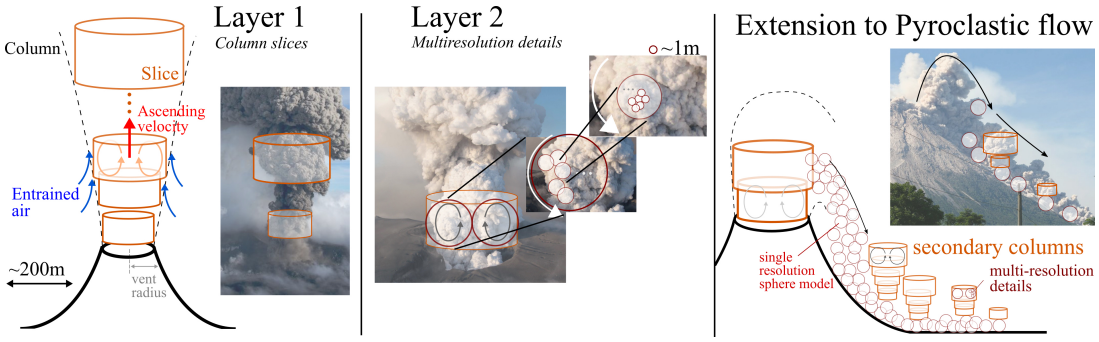

Figure 3.2: Our layered model makes use of circular slices to simulate and display the dynamics of the column (left), generates animated spheres at different resolutions to model small-scale visual details (middle), and provides an extension of these layers to handle pyroclastic flows (right). The pictures are annotated real photographs.

Dynamic simulation layer: The first layer aims at modeling the dynamics and global shape of the ejection column from medium scales (tens of meters) to large scales (several kilometers). This layer is grounded in volcanological studies (Spa86) which tell us that the local parameters in an ejection column are a function of the distance to the column's center-line and the altitude. Therefore, we represent the ejection column, at a large scale, as a generalized cone. As already explained, the turbulent flow entrains and heats the surrounding air, and consequently, the volume of the column increases as it rises. The key idea of the first layer is to simulate this coupled phenomenon at a medium scale by considering a series of radial, cylindrical slices representing a vertical sampling of the whole generalized cone (see Fig. 3.2-left). These slices are initialized near the volcano vent to the amount of material ejected during a given time interval, characterized by spatially averaged physical quantities (position, velocity, density, temperature). These values, as well as the volume and orientation of the slices, evolve over time, as will be detailed in Sec. 3.4. The radius and height of the slices are set to a common characteristic size r representing the radius of the largest vortex inside the slice. This size, initialized to the vent's radius r_0, increases with the raise of the column.

Procedural details layer: The coarse dynamic model we just described requires the addition of turbulent details, which we achieve through our second layer, a medium to small scales visual representation of the turbulent flow. In a highly turbulent flow as the eruptive column, multiple vortex rings are intricately mixed and their toroidal structure cannot be clearly distinguished. To represent this billowing motion, we reuse the idea of rotating spheres (Dre01; Kap03), by embedding the latter in column slices as shown in Fig. 3.2-middle. This representation, detailed in Sec. 3.5, efficiently captures animated visual details, up to the scale of a meter.

Pyroclastic flows can be animated using a simple extension of this layered model (see Fig. 3.2-right). Unlike the rising plume, these flows fall down along the terrain. We propose to reuse a single intermediate resolution of the second layer's sphere-based representation to compute the propagation of these flows. These spheres, seen as material particles, allow a simple and efficient computation of the interactions with the terrain. To model the possible formation of rising sub-columns where the mixture becomes buoyant again, We turn back to our first layer and emit new dynamic slices covering the areas where the mixture density is low. We then reuse our full column layered model on these new slices, with adapted initial conditions. I provide more details on this extension in Sec. 3.6.

3.4 Dynamics of volcanic columns

We use a Lagrangian simulation with discrete time-steps Δt to compute the dynamic evolution of volcanic eruption columns. As previously described, each simulated element is a cylindrical slice that represents a limited amount of ejected, gas, and ash mixture.

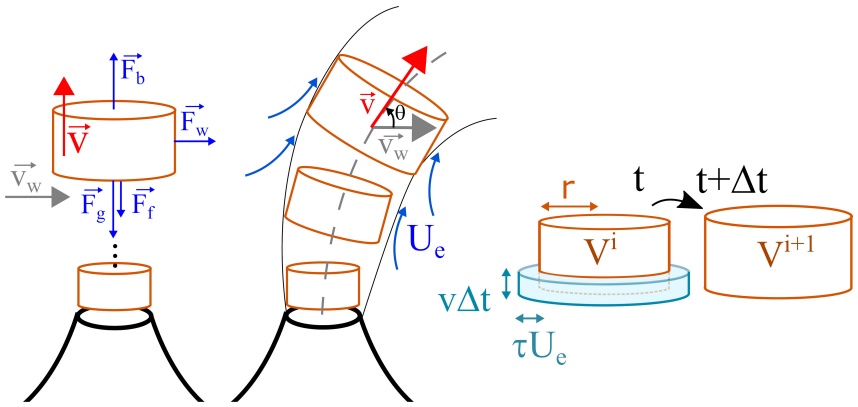

Figure 3.3: *Left*: Forces applied to a plume slice. *Middle*: Effect of wind on the path and orientation of slices. *Right*: Increase of slice volume at each time step, considering the entrained air represented by the blue ring.

3.4.1 Evolution of the ejection velocity

A new slice is emitted from the vent every $t_{slice} = r_0/(2\,v_0)$ seconds, depending on the emission speed v_0 and the vent radius r_0. The slice is parameterized using five independent quantities that vary over time: First, its characteristic size r (in m), which represents both the radius and height of the slice (vorticity ring with vortex of radius r) and gives its volume $V = \pi r^3$; Second, the position of its center $\vec{p}$, and in particular its altitude z (in meters); Lastly, the velocity $\vec{v}$ (in m/s), the density ρ (in kg/m^3), and the temperature T (in K) of the slice. In our Lagrangian representation, these three last parameters stand for averaged values of spatial distributions of the modeled quantities over the volume of the slice.

Since the mass $M = \rho V$ of a slice evolves over time due to entrained air, the fundamental equation of dynamics writes as $\mathrm{d}\,(M\,\vec{v})/\mathrm{d}t = \vec{F}$, where $\vec{F}$ is the sum of all forces applied on the slice (detailed in Sec. 3.4.2). Considering a finite variation of mass ΔM and velocity $\Delta\vec{v}$ during the time interval Δt, we get:

$$\Delta M\,\vec{v} + M\,\Delta\vec{v} = \Delta t\,\vec{F}. \tag{3.1}$$

Let us call x^i (resp. x^{i+1}, x^{i-1}) a value evaluated at the current (resp. next, previous) time step. We aim to discretize relation (3.1) with an explicit finite difference scheme expressing the updated velocity $\vec{v}^{i+1}$ with respect to the quantities computed from the current or previous time steps. This expression is not straightforward in the case of volcanic column dynamics, as the change of mass depends on velocity. To avoid an implicit, non-linear relation on $\vec{v}^{i+1}$ that appears in both $\Delta\vec{v}$ and ΔM, we inspire from symplectic Euler schemes and evaluate ΔM at the previous time step:

$$(M^i - M^{i-1})\,\vec{v}^i + M^i(\vec{v}^{i+1} - \vec{v}^i) = \Delta t\,\vec{F}^i\ , \tag{3.2}$$

which finally leads to the explicit relation:

$$\vec{v}^{i+1} = \frac{M^{i-1}}{M^i}\,\vec{v}^i + \Delta t\,\frac{\vec{F}^i}{M^i}\ . \tag{3.3}$$

We finally compute the new position by integrating the velocity.

3.4.2 Modeling the applied forces

The force $\vec{F}$ applied to each slice is split into four contributing components (see Fig.3.3-left), namely the weight $\vec{F}_g$, buoyancy $\vec{F}_b$, friction $\vec{F}_f$, and wind force $\vec{F}_w$. The slice weight is computed as $\vec{F}_g = M\ \vec{g}$, where $\vec{g}$ is the gravity constant. Buoyancy is set to:

$$\vec{F}_b = -\rho_a(z)\,V\ \vec{g}\ , \tag{3.4}$$

where ρ_a is the density of the atmosphere (outside of the volcanic column) at the slice altitude z. The variation of velocity between the column and the surrounding air induces a shear stress $\tau = \rho_a C_d v^2$ in turbulent flows, where $v = \|\vec{v}\|$ and C_d is a dimensionless drag coefficient. This stress induces a friction force acting on the lateral slice boundaries (in contact with the air), of area $S = 2\pi\,r^2$:

$$\vec{F}_f = -\frac{1}{2}\rho_a\,C_d\,S\,v^2\ \ \vec{v}/v\ . \tag{3.5}$$

Note that this force term is necessary for our bounded slice model, contrary to previous studies that relied on approximate radial velocity profiles tending to zero as r goes to infinity (MTT56). Finally, we model the force exerted by a horizontal wind of velocity $\vec{v}_w$, as:

$$\vec{F}_w = \lambda \, S_{\text{eff}} \, \|\vec{v}_w - \vec{v}_{|xy}\| \, (\vec{v}_w - \vec{v}_{|xy}) \ , \tag{3.6}$$

where $\vec{v}_{|xy}$ is the velocity component of the slice in the horizontal plane, $S_{\text{eff}} = 2r^2$ is the effective cross-section of the slice where the wind acts, and λ is a constant coefficient.

3.4.3 Evolution of a slice's size and density

The applied forces listed above depend on the size of the slice, i.e. its radius r from which surface and volume are computed, as well as on its mass. These quantities are evolving over time due to the entrainment of the surrounding air during the ascending phase. Let us describe our model for this evolution.

We rely on the notion of *entrainment velocity* (SK15) U_e, used in volcanic models to express, at the macroscopic scale, the averaged velocity at which the surrounding ambient air is incorporated and mixed within the turbulent plume. In the absence of wind, U_e is assumed to be proportional to the speed of the plume $U_e = k_s|v|$, where k_s is a *coefficient of entrainment*. In the case of wind of magnitude v_w, the relative contributions of the wind on the vertical and horizontal planes are modeled in volcanology as different and additive, leading to:

$$U_e = k_s \, |v - v_w \cos(\theta)| + k_w \, |v_w \sin(\theta)| \, , \tag{3.7}$$

where θ is the angle between $\vec{v}$ and $\vec{v_w}$ (see Fig. 3.3-middle).

This entrainment velocity is used at each time step to compute the newly incorporated volume of surrounding air within the slice. To this end, we consider such volume to be geometrically defined as a cylindrical ring with radius spanning the interval $[r, r + \tau U_e]$, and of height $v\,\Delta t$, giving an additional volume $\pi \, ((\tau \, U_e + r)^2 - r^2) \, v \, \Delta t$ as illustrated in Fig. 3.3-right.

The mix between the newly incorporated air and the current plume impacts the physical parameters of the slice, which are updated as follows: We first compute the new averaged temperature T^{i+1} assuming that the global heat capacity of the new mixture remains constant ($M_a^i = \rho_a(z) \, U_e^i \, v^i \, \Delta t^2$ being the mass of the entrained air, and $T_a(z)$ a standard normalized temperature model of the atmosphere at altitude z):

$$T^{i+1} = \frac{M_a^i T_a(z) + M^i T^i}{M_a^i + M^i} \ , \tag{3.8}$$

This change of temperature results in a change of slice volume, from V^i to $V^{i'}$ for the slice and V_a to $V_a{}'$ for the atmosphere annulus, and obtained from the perfect gas equation: $V^{i'} = V^i\ T^{i+1}/T^i$, and $V_a' = V_a\ T^{i+1}/T_a(z)$. The new volume of the slice is the sum of these two new volumes, and the new radius and height, set to the same value r^{i+1}, are computed from the equation of a cylinder's volume as $r^{i+1} = \sqrt[3]{(V^{i'} + V_a{}')/\pi}$. In addition to the entrainment of air, the mixture of the slice also slowly sediments, leading to the loss of the heaviest particles that fall down and leave the slice. We model this phenomenon by assuming a constant decrease of mass σ_{col} over time. The updated total mass of the slice is, therefore:

$$M^{i+1} = M_a^{i+1} + M^i - \sigma_{col}\Delta t\,. \tag{3.9}$$

and, the new slice density can be computed as $\rho^{i+1} = \frac{M^{i+1}}{V^{i+1}}$.

Independently from these parameters, the slices are always oriented orthogonally to their trajectory during their ascending phase, in accordance with the assumption that each slice contains a large vortex ring constrained to translate orthogonally to the central axis.

3.5 Procedural details and umbrella

3.5.1 Multi-resolution, rotating spheres

As explained in Sec. 3.3.2, billowing volcanic flows take the visual appearance of rotating spheres at multiple scales, of expanding sizes during ascent. In this section, I detail our second layer, which models this rich visual appearance, using procedural spheres guided by simulated slice dynamics (see Fig. 3.2-middle).

At the largest scale s_1, the rotating spheres have a radius similar to the characteristic length r of a slice and exhibit a convective motion relative to the main averaged ascending movement. More precisely, the convection drives a toroidal motion (upward velocities close to the central axis of the column, downward at the boundaries), which can be represented by vortex rings (CK00). To imitate the toroidal motion, we associate to each slice a vortex ring, defined by an inner circle with same position and orientation as the slice, and with radius $r/2$. We then uniformly sample n overlapping spheres of radius $r/2$ so that their center lies on the inner circle, and perturb the position and scaling of the spheres by a small factor to achieve natural variations. The velocity of the spheres is set to the slice velocity $\vec{v}$, while the angular velocity $\vec{\Omega_s}$ is given as a boundary condition enforcing the velocity at the exterior of the column to be equal to the air velocity: $\vec{\Omega_s} = v_s/r_s\vec{t_s}$, where $\vec{t_s}$ is the unit vector tangent to the inner circle and oriented clockwise.

The surface of the spheres in s_1 is sampled by a series of recursively defined, smaller spheres, in

order to model the cascading vortices of the turbulent flow. Based on observation, we typically consider two more layers of spheres s_2 and s_3, of radius decreasing by a factor five from a sphere to its parent. In our experiments, this leads to a size of about a meter for the smallest visible details, generated by the smallest s_3 spheres located near the volcano vent. Each child sphere is initialized at a random position on its parent sphere and is rigidly attached to the motion of the latter. This allows to better perceive the general rotational motion at scale s_1.

3.5.2 Procedural umbrella over the plume

When the altitude $z_{neutral}$ of equilibrium density is reached, the pressure from the underlying rising column, which continuously brings new material, leads to a horizontal spread of the flow. The latter then takes a characteristic *umbrella* shape.

Our model for animating this umbrella is twofold: First, at the end of the ascending phase, the column trajectory slightly *over-shots* beyond the neutral buoyancy altitude $z_{neutral}$ due to inertia. This phenomenon, which can be observed in real eruption footage, is already captured by the dynamic simulation used for slices. Second, once this *overshot* maximal altitude is reached, we *detach* the large scale spheres from their associated slice, and procedurally adapt their individual trajectories to animate a smooth transition from ascending motion to horizontal spreading.

To this end, we compute radial motion as to ensure volume conservation: we consider that a slice reaching $z_{neutral}$ with a vertical velocity v then expands radially with the same velocity. During an infinitesimal time ∂t, the volume of this slice (initially defined by $V = \pi r^3$) then increases by $\partial V = \pi r^2 v\, \partial t$. To accommodate for this expansion, all material at distance d from the central axis of the column needs to increase by $\partial V = \pi r\, dv\, \partial t$, leading to a new velocity $v_{umbrella}(d) = v\, r/d$ for the spheres, d being the radial distance to the center of the rising column. Lastly, we generate a smooth asymptotical descending motion for each sphere toward their final altitude $z_{neutral}$, using a procedural decreasing function shaped as $1/d$.

3.6 Modeling pyroclastic flows

Depending on the initial density and ejection speed, the volcanic column may not reach the critical amount of buoyancy to compensate for its weight. In this case, the ash mixture falls down shortly after its ejection and leads to a pyroclastic flow.

Our model switches to pyroclastic behaviour during simulation (after the evaluation of Eq. (3.3)), if a slice's velocity becomes negative before reaching the neutral buoyancy altitude. In this case, we trigger specific pyroclastic effects, which requires a few modifications of our layered model. Two phenomena

need to be modeled. First, the propagation of the descending ash and gas mixture down the terrain slope, and second, the possible formation of ascending sub-columns, due to the loss of density of the mixture related to ash sedimentation (see Fig. 3.2-right).

We rely on the four forces already described in Sec. 3.4.2 (weight, buoyancy, friction, and wind force) to simulate the dynamics of the pyroclastic flow. However, since the pyroclastic flow descends the slope of an arbitrary terrain, its complex geometry cannot be captured by our cylindrical slice model. Therefore, we replace the falling slice geometry with a set of falling spheres initialized to uniformly fill the slice's volume. In practice, we use the scale s_2 of our sphere layers, to provide enough degrees of freedom.

Each sphere is assumed to be an independent particle simulated under the actions of weight, buoyancy, friction, and wind forces. The expression of these forces is adapted to the geometry of the spheres, whose volume is supposed to remain unchanged, despite sedimentation, during flow propagation. The interaction with the terrain, modeled using collision with a height field surface, allows to efficiently model the spread of the flow along the slope. During this spread, the mixture sediments the heaviest particles, which we model as a constant mass decrease over time, by a factor σ_{pyr}.

As the mixture loses weight and density, the simulated spheres may become buoyant and elevate again in the air. We use a clustering method to identify groups of near-by buoyant spheres with relative distance smaller than r_{pyr} that will trigger the creation of a rising column using the method in Sec. 3.4, but with different initial conditions. The clustered spheres are replaced by new rising slices starting from their centroid, with a characteristic size r_{pyr}. To preserve mass and momentum, the initial density and velocity of the slices are set to the average of those of the clustered spheres, which are then removed. Once created, the new columns follow the same simulated model as the primary column and can be pushed by the wind. Note that the pyroclastic flow progressively brings new buoyant spheres, which are converted to slices added to the volume of the rising sub-columns, eventually over a large area. As opposed to vent ejection which (a continuous and localized flow source), each slice creation in a pyroclastic flow happens at a given time, but the process repeats at various ground positions.

3.7 Visualization and results

3.7.1 Real-time animation and rendering

We implemented a real-time graphics prototype as a single thread C++ code with OpenGL display and run it on a consumer laptop (Intel Core i7 2.6GHz CPU, NVidia Quadro P3200 GPU, 16GB RAM). We provide the open-source code of this prototype here: https://gitlab.com/mlastic/volcanicplume.

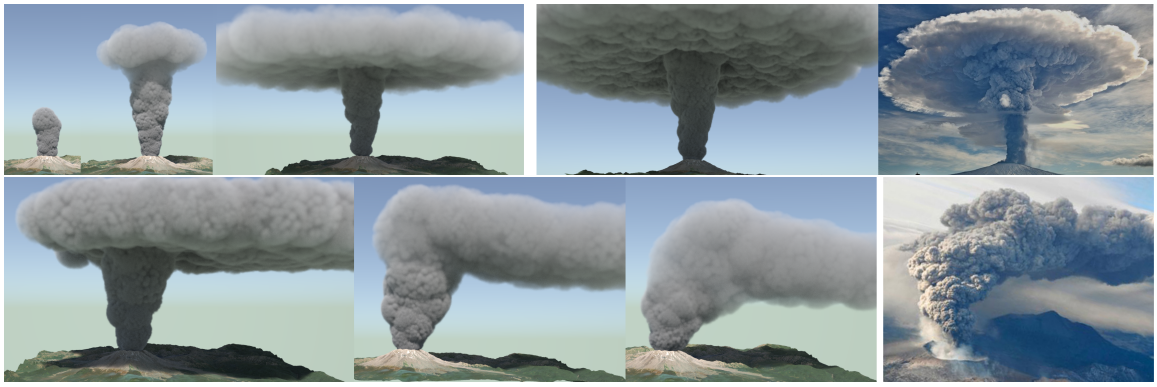

Figure 3.4: Top: Evolution of a rising plume through time and its spreading umbrella shape in the atmosphere (off-line rendering) compared to a real photograph at the right. Bottom: Effect of an increasing wind of 15, 45, and 105 m/s on the column profile (off-line rendering) compared to a real photograph.

Visualization: As illustrated in Fig. 3.5-left, our prototype allows to visualize the slices and spheres, as well as to convey a fuzzy appearance using billboards. In the latter, smoke-like texture is mapped on quadrangles facing the camera and following the center and dimension of the spheres, while their angular speed in the camera plane is expressed as the projection of the 3D angular velocity of the slice onto the forward direction of the camera.

Performance: Although not optimized, our implementation runs at more than 60 fps for all our examples. We used up to 80 slices and visualized the two first sphere scales. If needed, higher levels of details could be displayed, but may need acceleration using graphical tessellation or instancing.

Model parameters: Table 3.1 provides the default values for our parameters. r_0, ρ_0, and T_0 correspond to standard volcanic measures. Our empirical, drag coefficient C_d matches the expected order of magnitude for the standard aerodynamics model. We used values from (WHPS13) for (k_s, k_v). The other coefficients were tuned empirically.

Simulation time step Δt 0.002s	Initial density ρ_0 200kg/m^3	Vent radius r_0 100m
Initial temperature T_0 1273K	Drag coeff. C_d 0.04	Wind force coeff. λ 600
Entrainment coeff. (k_s, k_v) (0.09,0.9)	Air entrainment coeff. τ 5s	# spheres/slice n 6
Column sedim. coeff. σ_{col} 5×10^{-8}	Pyroclastic sedim. coeff. σ_{pyr} 5×10^{-6}	Secondary col. size r_{pyr} r_{s_2}

International Standard Atmosphere model
$\rho_a(z) = 353 \frac{(1-2.3\times10^{-5}z)^{5.3}}{288-6.5\times10^{-3}z}$, $T_a(z) = 288 - 6.5 \times 10^{-3}\, z$

Table 3.1: Parameters used in our simulation.

Our model allows the user to dynamically tune them at run time. For instance, the emission velocity

and density, the wind direction and orientation at a different angle, or even the vent radius, can be easily changed. Such interactive control on simulation parameters, coupled with real-time computations, is essential to explore various behaviors for educational purposes, or in virtual museums. It can also be used for quickly testing new hypotheses in volcanology applications before running a costlier simulation on a restricted domain.

3.7.2 Validation of Column Dynamics

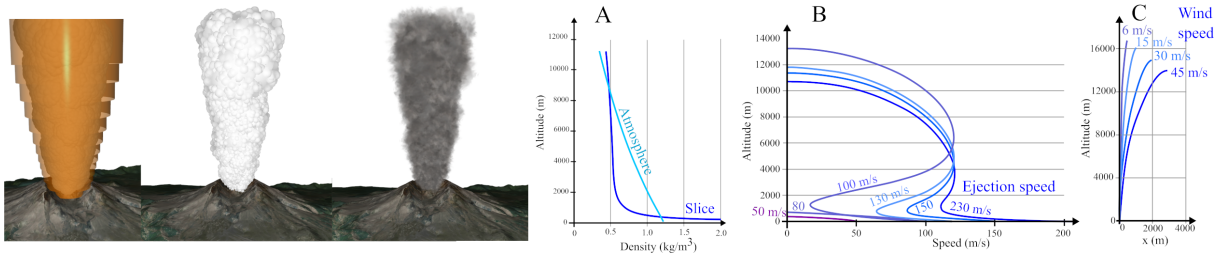

Figure 3.5: Left: Real-time previsualization of the column dynamics showing respectively the slices; the first and second scale spheres; rotating billboard. Right: Validation of the evolution of the simulation parameters. A: Evolution of the density of the column with respect to the altitude, compared to the density of the standard atmosphere (for a given ejection speed and no wind). B: Evolution of the altitude and vertical speed of a slice for different ejection speeds. C: Trajectory of a slice displayed in a 2D graph when a constant horizontal wind is applied. Stronger wind velocity leads to a lower maximal altitude.

We analyzed the numerical results of our column dynamic model and plot in Fig. 3.5 curves that can be found as references in volcanology literature. Fig. 3.5-A shows the evolution of the density of a slice ejected at an average velocity of 150m/s with respect to its altitude. As expected (Woo88; MDN02), the density decreases quickly at low altitude, and cross a first time the atmospheric density to become lighter than the surrounding air. The slice then becomes propelled by buoyancy until reaching 9km where the slice density and the atmosphere reach again an equal density. The slice slightly exceeds this altitude due to inertia and ends up reaching a final equilibrium. Fig. 3.5-B represents the evolution of the altitude of a rising slice with respect to its vertical speed. The multiple curves displayed correspond to different initial velocities at the vent. These curves are consistent with the one reported by Carey et al. (CB15) (Fig.32.3) with respect to their general shape and behavior relative to the emission velocity. In the case of an emission velocity below 80 m/s, a quick decrease of the vertical speed is obtained as the mixture doesn't get buoyant and falls back as pyroclastic flows. For higher initial speed, after the initial slow down, the column gains temporary speed due to the buoyancy propelling at low altitude, which leads to the noticeable inflection of the curves also reported in (CB15). Finally, the velocity slowly decreases near its maximal altitude. Fig. 3.5-C reports the trajectory appearance of the central line of the column when winds with various magnitudes are applied. For these measurements, we consider wind profiles that linearly increase from a zero magnitude on the ground to the maximal specified value at

12 km height. The curves we obtain can be compared to the one reported in Woodhouse et al. (WHPS13) (Fig. 2) relative to the Eyjafjallajökull eruption in 2010 using the same wind profiles, with agreement on the characteristic bent trajectory, as well as the fact that higher wind velocity leads to lower maximal altitude.

3.7.3 Off-line Rendering

Once the user is satisfied with the real-time simulation, the spheres data can be exported to an external volume-based renderer. We use Houdini (Sid21), which is set up to convert a set of spheres to an implicit surface. We create a 500^3 voxel grid and store the distance to the surface (using a Signed Distance Function, or SDF) in a narrow band of 6 voxels around it. In this band, we consider a smooth *visual-density* used by the Houdini's built-in volumetric renderer, while taking into account the two following features: First the per-sphere material density ρ that decreases with elevation, capturing the smoother and less dense appearance for the ejection flow at high altitudes. Second, we ensure a smooth rendering of the flow boundary by rescaling and clamping the SDF so that the resulting *visual-density* is 1 inside the plume and smoothly transitions toward 0 at the exterior. We then multiply this *visual-density* with a lower-dimension one, computed on a volume of (100^3) where we sub-sampled the per-sphere material density. We finally add small details below the scale of the smallest spheres using a cellular Worley noise. In practice, our off-line rendered scenes use 5 to 20 million individual spheres to generate the required density.

3.7.4 Final results

We simulated and rendered animations for several scenarios. Fig. 3.4-top shows the evolution of a rising plume with an ejection speed of 150 m/s and no wind. We can see the gas and ashes mixture spread symmetrically from the central plume when reaching the neutral buoyancy altitude, in agreement with real photography shown in the right part. In comparison, Fig. 3.1-middle, Fig. 3.4-bottom shows the result of a similar ejection speed, but in the presence of left-to-right wind. The umbrella spreads slightly non-symmetrically under a weak wind, while its propagation becomes almost unidirectional under stronger winds. Fig. 3.6 shows a time-lapse of a pyroclastic flows animation. In this case, the ejection speed of the column is about 70 m/s, which is too low to enable the ejected column to become buoyant. The mixture then falls back along the slopes of the volcano (three first pictures). In the last picture, sub-columns start to raise from spheres clusters that lost enough weight to become less dense than the air.

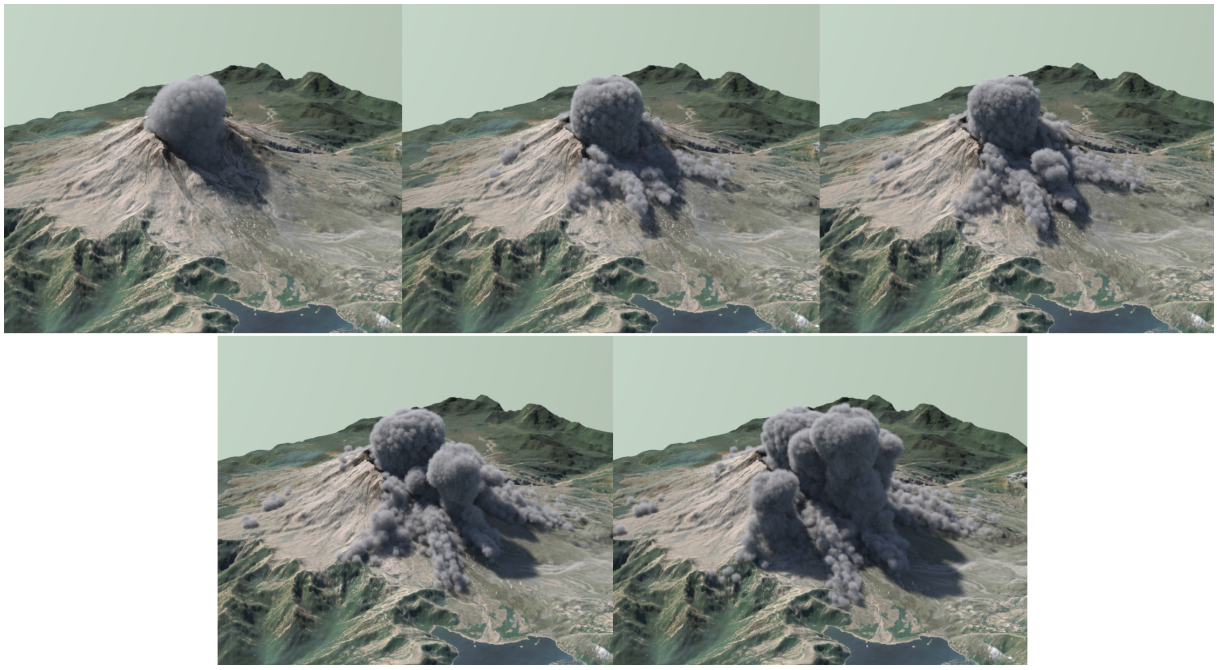

Figure 3.6: Pyroclastic flows animation: from left to right, first the top row and then the bottom row, the column begins to fall down; the pyroclastic flows go down the slopes of the volcano; secondary columns appear above the flows; secondary columns rise (off-line rendering).

3.8 Conclusion and future work

We presented a layered model for the real-time animation of volcano ejection. The first layer of our model is a lightweight Lagrangian simulation based on a discretization of the rising column into slices. This combination of cylindrical geometry and dynamic simulation allows handling physical phenomena that were never modeled in CG but are at the core of ejection dynamics such as air entrainment, drastic density variations, and volume expansion over large distances. Our dynamic model also triggers the automatic switch between jet and buoyancy propelling. The scale of the slice, which expands over time, represents the scale of the largest vortices at a given height, and can therefore be easily visually enriched while preserving real-time performances, using a second layer. The latter is composed of a multi-scale set of spheres representing the billowing aspect of the turbulent flow. We also reused these two layers to represent pyroclastic flows, where a simple sphere model is used to compute the spread of the ejection along the terrain slope, while sub-columns are generated using the previous Lagrangian simulation where and when the mixture becomes buoyant again. While remaining an approximation, our model is consistent with respect to real volcano dynamics, as shown by our comparisons of velocity and density values over time with reference data from volcanology literature. In addition, it brings plausible visual results, which can be pre-visualized in real-time to ease parameter tuning.

Our layered model suffers inherent limitations since each element is designed to match some ex-

pected behavior. Such lightweight model is a trade-off between realism, speed, and control, and is restricted to the actual phenomena it was designed for, here ejections from explosive volcanic eruptions. This limits its usability in other types of eruptions, *e.g.*, effusive ones where lava flows are to be combined with smoke.

Our current prototype could however be easily improved: The Kolmogorov cascade of details modeled by the spheres could be enhanced by adding sub-motions at the lower scale, and by using different sizes for the smaller spheres. Faster volumetric rendering and adapted shaders available in standard game engines could also be used to improve the rendering of the interactive simulator.

As future work, an avenue for research, made easier by our layered model, would be its inverse simulation, enabling to enrich artistic control by pre-setting or even sketching characteristic features of the ejection such as maximal elevation, column shape, or pyroclastic spread, with some automatic tuning of simulation parameters.

In this chapter, we use a theoretic atmosphere, with theoretic formulas giving the temperature and pressure depending on the altitude, and a simple linear wind. The next step would be to couple our volcanic eruption model with lightweight atmospheric simulation, to visualize the condensation phenomena associated with the fast rising motion of entrained, humid air. Adding the visualization of ash rain and lightning would also improve the realism of our results. We tackle this issue in the next chapter of this thesis, presenting in the first sections a lightweight atmospheric model I contributed to, and how we merged the two models for the plume and atmosphere in the last section of Chapter 4.

Chapter 4

Simulation of clouds and volcanic plumes in the atmosphere

4.1 Introduction

Volcanic eruptions are not only spectacular and devastating natural phenomena on their own, but also because of the impact the may have on weather. Close to the volcano, they have an effect on weather since the affect local cloud formation, they can cause rain and a change of temperature. They may also change the global weather of the Earth during months or even years for the largest eruptions. Indeed, water vapor and ashes are ejected in big quantities but they only have a local short-term effect on the weather, while other ejected particles such as sulfur dioxide stay for longer in the atmosphere and can cause of cooling of the Earth and specific colouring of the sky. A famous example is the huge eruption of the Krakatoa in Indonesia in 1883 which caused a global cooling of the Earth of 0.25°C. In addition it produced a red coloring of the sky all over the world in the following years, which is why the sky is red in some paintings of that time like *The Scream* of Munch. Coupling an atmospheric model with our volcanic eruption system would be interesting to model such effects.

Before explaining this coupling at the end of this chapter, I will first present the atmospheric model I contributed to at the beginning of my PhD. This model let us animate clouds by simulating the weather.

The key insight of this work is to decouple large-scale, versus small-scale features. We model the former using a coarse physically-based simulation, which yields plausible wind and cloud shapes while remaining tractable, and couple it with the plume system described in Chapter 3, which is more or less the same scale. In contrast, small-scale features such as precise cloud appearance, are generated procedurally based on the output of the simulation. This enables us to maintain a coherent appearance across

scales, while achieving inexpensive computation compared to a simulation of equivalent resolution.

Our technical contributions include:

- A layered model of the atmosphere enabling the independent tuning of horizontal and vertical resolutions, and an associated fluid thermodynamics simulation method, which efficiently captures both intra-layer advection and inter-layer convection;
- A practical solution for modeling meteorological interaction with terrains of varying topography and surface coverage;
- A mechanism for classifying clouds based on a standard taxonomy that enables the coherent upscaling of our simulation results.
- An extension of both the weather and the plume models by coupling them, leading into a more complete model able to generate more types of clouds and ash rain.

The next section of this Chapter has led to a publication (VGL+20).

[Ulysse Vimont, James Gain, Maud Lastic, Guillaume Cordonnier, Babatunde Abiodun, and Marie-Paule Cani. Interactive meso-scale simulation of skyscapes. CGF, Proc. Eurographics, 39(2), 2020.]

My personal contribution on this project was on the simulation part: I improved and expanded the work Ulysse Vimont had begun on the meteorological framework, and I set up the interaction with the terrain.

The last section about the extension to weather and plume coupling is still an ongoing project, and the results presented here are not the final ones.

4.2 Weather model

In this section, I present the weather simulator we developed to animate clouds and skyscapes, and which will be later coupled with volcanic plumes.

4.2.1 Motivation

Weather, and its impact on the skyscape, plays a key role in the digital depiction of outdoor landscapes. Both the sweep of the sky and the way it scatters light and shadow across natural elements are crucial to the mood of scenes in games and feature films. Moreover, wind and clouds are even more critical to specific applications, such as flight simulators.

Although extensively investigated in Computer Graphics, physically-realistic clouds cannot yet be generated and animated at interactive rates. Despite advances in the modeling of cloud-like shapes,

Figure 4.1: A $50 \times 50 \times 10$km skyscape obtained from our simulation, showing multiple layers with different cloudforms.

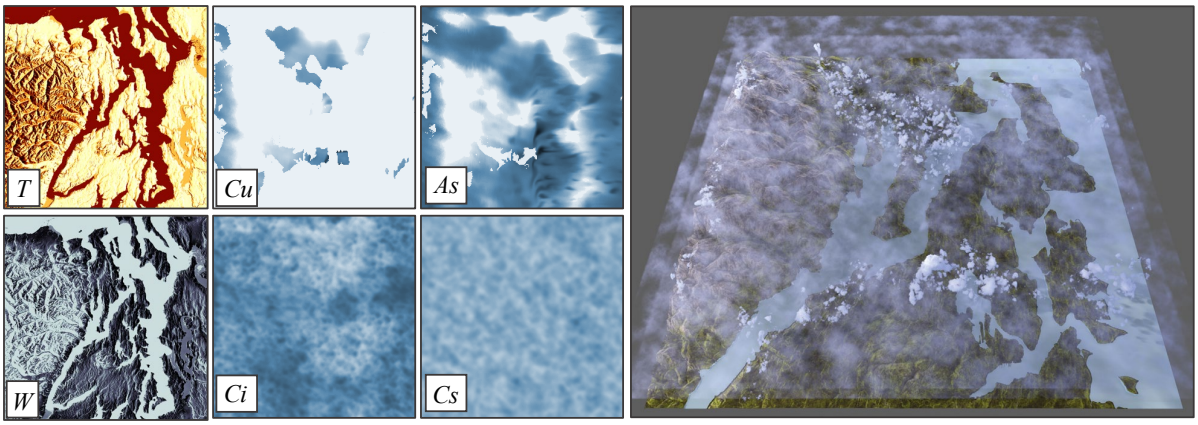

Figure 4.2: (Left) Our meso-scale simulation engine generates temperature (T) and water (W) terrain data, as well as cumulus (Cu), altostratus (As), cirrus (Ci) and cirrostratus (Cs) cloud layers. (Right) These sparse outputs are procedurally amplified to enable the rendering of a detailed volumetric skyscape.

automatic detail enhancement and volumetric rendering, no solution exists for the efficient and detailed simulation of wind effects and attendant cloud evolution. Procedural methods are prevalent in practice and rely on user specified key-frames with inbetweening through linear blending or optimal transport. Despite recent improvements, these techniques are unable to generate the full variety and complexity of dynamic skies observable in nature.

Cloud development is governed by specific physical laws and processes (WH06). Clouds are a physical manifestation of condensation of water vapour in air. They form when a moist air parcel reaches a saturation point, either by cooling to its dew point temperature or by evaporating water. Cloud formation and type depend on atmospheric conditions, as well as the structure of the underlying terrain. For instance, atmospheric temperature and moisture profiles, evaporation from water bodies, and interaction between the wind and mountain slopes, dictate the formation and shape of clouds over time.

To fully simulate these phenomena is a daunting prospect. The physical processes involved, including thermal and fluid dynamics, are complex and tightly coupled. The simulation domain needs to be both large in extent and finely sampled to adequately capture cloud structure and interactivity is difficult to achieve given the computational demands.

We propose a new layered atmospheric model that supports efficient simulation and can be coupled to simulations of other local phenomena such as volcanic eruptions, combined with a mechanism for consistent rendering through detail amplification based on cloudform classification. Although our layered representation is vertically sparse, the attendant simulation captures the underlying physics behind wind and cloud formation, such as interactions between atmosphere and terrain. Our framework also yields visually plausible results thanks to a consistent, procedural volumetric up-sampling mechanism: simulation outputs such as wind velocity, convection and altitude are used to detect cloud-type, enabling the derivation of consistent noise parameters for cloud density. Finally, volumetric rendering can be applied directly to the up-sampled density datasets.

Our framework not only captures the variety of cloud types that are visually salient in nature, but also incorporates other important meteorological effects. This is achieved at meso-scales (up to $50 \times \times 50 \times 10$km) with a level of resolvable cloud detail suitable for virtual environments. The simulation is run at near interactive rates($< 10s$ per frame) using accelerated time scales (where each time-step represents from 30 seconds to 1 hour of the final real-speed animation), enabling the user to iteratively author a variety of meteorological scenarios at coarse scales. The results can then be refined both spatially and temporally for final rendering. As our results show, the simulation seamlessly captures both stratiform and convective cloud formations (see Figures 4.1 and 4.2). Furthermore, our model accounts for physical effects, such as phase changes (evaporation and condensation), latent heat exchange between ground and atmosphere, and orographic uplift, where mountains direct the air upwards.

4.2.2 Related Work

Category	Method	Cloud Type	Scale	References
Procedural	Noise	Stratiform	Meso	(BNL06)
	Implicit Shapes	Convective	Micro, Meso	(Ney97; BN04; LJW06; YLH+14)
	Noise & Implicit Shapes	Mixed	Meso	(Gar85; WCGG18; SSEH03)
Simulation	Eulerian	Mixed	Micro	(DKY+00; MYND01; MIY02)
	Eulerian	Mixed	Meso	(GDAB+17)
	Lagrangian	Convective	Micro, Meso	(Ney97; WBDY15; FBDY15; DG17)
Hybrid	Lagrangian & Noise	Convective	Meso	(GN17)
	Eulerian & Noise	**Mixed**	**Meso**	**Ours**, , (HMP+20; HHP+21)

Table 4.1: A comparison of cloudscape approaches, comparing their underlying method, achievable cloud type (stratiform, convective or a combination of both), simulation scale (micro up to 1km, meso up to 100km and synoptic scales beyond) and research representative of each category.

Related work about weather and cloud animation has already been presented in Chapter 2. The table 4.3 sums up the main previous methods divided into procedural, simulation, and hybrid models, and compares the work presented in this section to them.

4.2.3 Meteorological Framework

4.2.3.1 A Primer on Meso-scale Meteorology

A skyscape (also sometime called a cloudscape) is the vista that can be seen from a given viewpoint, often located on the ground. Its visual appearance can undergo dramatic change over the course of a single day due to day-night cycles and meteorological phenomena designated as *weather*.

Weather takes place in the troposphere, the lowest 10km of the atmosphere that contains more than 75% of its mass. The upper boundary of the troposphere is the tropopause, where a strong increase in temperature with height prevents any ascending air movement. The lower boundary of the troposphere is the earth's surface (ground or water body) with which the troposphere exchanges heat, moisture and momentum. Most of the energy that drives weather is from the sun and reaches the atmosphere and earth surface in the form of radiation. The ground receives almost all the solar energy and transfers it back to the atmosphere through radiation as well as conduction to the lowest layer of air. In addition to heat, the ground also exchanges moisture through evaporation and condensation. However, these transfers induce energy discrepancies between air parcels at different locations, which in turn fuels complex air movements in both horizontal (wind) and vertical directions (thermals). Air further from the surface is cooler due to less pressure being exerted by the remaining air above it. Warm and cold air are hardly miscible, and the differences in temperature lead to a stratified structure in the atmospheric temperature profile. Present in most circumstances, this horizontal stratification of air becomes more obvious when a sea of clouds or a low-altitude smog trap appear due to the presence of an inversion layer (an air layer in which temperature increases with altitude).

When energy discrepancies are sufficient to force moist air to rise above its lifting condensation level (the altitude where relative humidity reaches 100%), vapor will condense and clouds appear. The surface of a cloud follows the general air motion and is therefore a good indicator of the origin of the rising air. Convective motion is very turbulent due to the internal and boundary velocity discrepancies and tends to create cumuliform clouds, whereas synoptic wind forcing air above a relief or a colder air mass tends to create stratiform clouds.

Water droplets tend to coalesce, increasing their weight to surface ratio until vertical air motion is too low to maintain them in the air: this is precipitation. This motion of the water being evaporated, transported, and precipitated is known as the water cycle. It is more complex that the explanation given

here, but out of the scope of this paper. For further details on meso-scale atmospheric physics, we refer the reader to works by Barry *et al.* (Bar92; BC09) and Pielke *et al.* (PSA+98).

In summary, generating a descernible skyscape where the different cloudforms are visually distinguishable requires capturing fine-scale (~10m) cloud detail in an horizontal span up to $50\ km^2$, with a vertical extent of the entire troposphere ($10\ km$). The resulting scale ratio (~10^{10}) makes this ill suited to computer memory storage and processing. In the subsequent section, we present a strategy for dramatically reducing the required memory footprint of a skyscape representation, while still allowing a physically-based simulation of its temporal evolution.

4.2.3.2 Method Overview

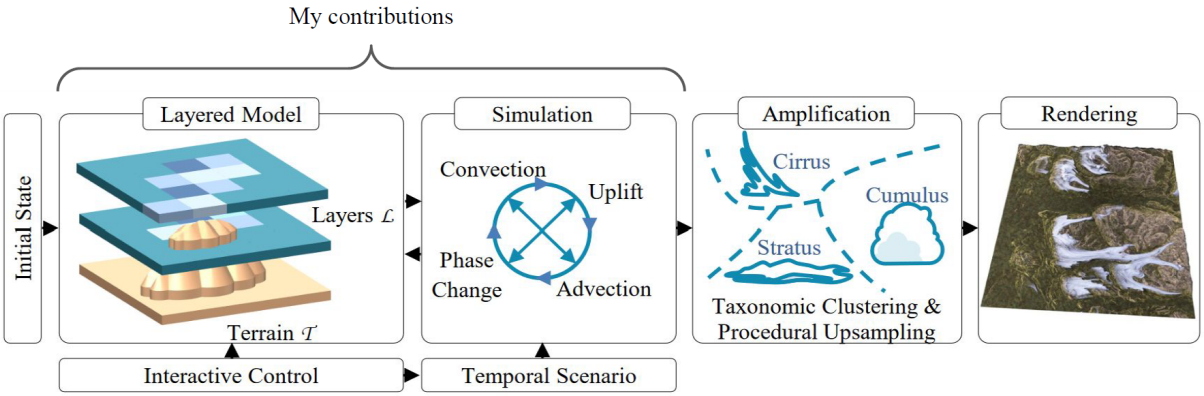

Figure 4.3: Overview: A large-scale layered atmospheric model is updated at interactive rates by a meteorological simulation. The later can be used to feed a fine-scale procedural volumetric amplification process, used for the offline rendering of animated skyscapes. I highlighted my contributions on this model: I worked on the terrain interaction and on the meteorological framework with Ulysse Vimont.

Our general approach (as outlined in Figure 4.3) is to apply meteorological simulation over a sparse set of atmospheric layers at meso-scales (approximately 200×200m horizontal resolution per cell) followed by noise-based upsampling keyed to quantifiable attributes, such as convection, advection, and altitude, in order to achieve micro-scale volumetric rendering.

Our layered atmospheric model and matching simulation (Section 4.2.4) are motivated by the stratification evident in meso-scale meteorology. For a given skyscape scenario a sparse irregular stack of horizontal layers is constructed, with each layer structured as a 2D regular grid of meteorological data. This serves the needs of both modelling and implementation: layers can be situated to best capture meteorological features, while their overall sparsity improves simulation efficiency and the individual layers themselves are a good fit for GPU texture memory.

We tailor our meteorological simulation to this novel representation, in order to consistently evolve layer data over time at interactive rates. Motion within layers is handled through an extension of stan-

dard 2D Eulerian divergence-free computational fluid dynamics (And18) to encompass atmospheric quantities, such as temperature, absolute humidity, and water content. Motion between layers is divided into two categories: convective motion modeled trough a difference in moist static energy and non-convective motions due to the orographic effects of terrain.

The handling of terrain requires special consideration (Section 4.2.5). Not only are there various heat and moisture fluxes between land and air, such as irradiation, evaporation and condensation, but topography also shapes winds and introduces orographic effects. We represent terrain as a height-field that may cut through one or more atmospheric layers, with horizontal advection and vertical convection being adapted accordingly.

As a last step we apply procedural amplification to increase volumetric detail preparatory to rendering (Section 4.2.6). I won't go too much into details in this part since I did not work myself on this section, as I was focused on the simulation process. Cloud forms are governed primarily by a combination of altitude, turbulence due to convection, and wind strength. These quantities are readily available in our simulation and can be used to assign a cloud type according to the standard taxonomy (see Figure 4.5). Then simulation cells are upsampled using harmonic noise with cloud-type-specific parameters. Finally, at each simulation step, the scene is rendered using ray-marching of the procedurally-amplified volume.

4.2.4 Efficient Atmospheric Simulation

4.2.4.1 Layered Model

As already stressed, clouds in nature are structured in horizontal layers and atmospheric quantities, such as temperature, do not decay linearly with altitude. Thus, even if a regular vertical sampling (and the attendant $3D$ voxel grid) did not suffer from serious memory overheads, it would still be a poor fit to the meteorological context. This was already recognized by Garcia-Dorado *et al.* (GDAB+17), who uses voxels of exponentially increasing thicknesses. Furthermore, the merits of such uneven atmospheric sampling are recognized in the meteorological simulation literature (RBD+18), albeit in a different application context. We generalize this by representing the atmosphere as any irregular stack of horizontal $2D$ layers.

A benefit of our approach is that we can pack each layer into a texture and exploit GPU acceleration to perform $2D$ within-layer advection of temperature and moisture. Vertical between-layer transfers must account for the uneven spacing between layers and are handled separately.

In our method, the user specifies the number of atmospheric layers (n), the layer resolution, and the altitude of each layer. For simulation purposes, each of the atmospheric layers $\mathcal{L}_i$, $i = 0, \ldots, n-1$, is then encoded as a grid with the following data fields (see Table 4.2): *temperature* T_i in Kelvin reflecting

Category	Notation	Quantity	Units
Atmospheric layer $\mathcal{L}_i$	Θ_i	altitude (constant)	m
	T_i	temperature	K
	H_i	absolute humidity	g/m^3
	W_i	water content	g/m^3
	$\mathbf{V}_i$	2D velocity	m/s (2D)
	U_i	updraft velocity	m/s
	C_i	convection velocity	m/s
	D_i	vertical velocity	m/s
	P_i	dynamic pressure	Pa
	Π_i	atmospheric pressure	Pa
	σ_i	orographic velocity damping	none
Terrain $\mathcal{T}$	$\Theta_\mathcal{T}$	altitude	m
	$\Lambda_\mathcal{T}$	heat capacity	$J/kg/K$
	$T_\mathcal{T}$	temperature	K
	$M_\mathcal{T}$	water content	g/m^2
	ε	emissivity	none
	B	albedo	none
Coefficients	λ	convective coupling	$m.s^{-1}.J^{-1}$
	ν	non convective coupling	$m.s^{-1}.Pa^{-1}$
	μ	pressure coupling	$Pa.s$
	γ	humidity transfer rate	s^{-1}
	κ	heat transfer rate	$K.J^{-1}$
	Δt	time step	s

Table 4.2: Symbols used in this paper.

the mean temperature of a cell; *absolute humidity* H_i in g/m^3, which is the quantity of water vapour (uncondensed moisture) present in a cell; *water content* W_i indicating the quantity of condensed (liquid) water in g/m^3 and *2D velocity* $\mathbf{V}_i$, which is a measure of the local planar wind direction and strength and is used to advect the other quantities.

4.2.4.2 Meteorological Simulation Engine

Our simulation engine is built on the advection of atmospheric quantities using standard computational fluid dynamics (CFD), with two significant amendments. First, we account for phase changes between the liquid and vapour states of water, including the attendant release and absorption of latent heat. Second, we transfer quantities vertically between adjacent layers in order to simulate convective and dynamic uplift arising from vertical gradients in temperature, moisture, and pressure.

At each simulation timestep we apply $2D$ Eulerian CFD (And18) to each atmospheric layer $\mathcal{L}_i$, by computing a dynamic pressure field P_i using the Poisson equation:

$$\nabla^2 P_i = \nabla \cdot \mathbf{V}_i. \tag{4.1}$$

This is solved in the classical manner using a multi-grid method. Next, the pressure field is used to make the velocity field divergence free (∇P_i is subtracted from $\mathbf{V}_i$). In typical CFD, the velocity and a generic concentration field would then be advected. Instead of working on anonymous concentration

we advect temperature T_i , absolute humidity H_i , and water content W_i in addition to velocity $\mathbf{V}_i$. This is not a significant point of departure: advection of such quantities is not uncommon in Eulerian cloud simulations (MYND01; MIY02; GDAB+17).

Phase changes: Our first substantive addition to the Eulerian scheme is to incorporate evaporation and condensation within layers. These occur when the saturation humidity H_i' for a given cell is above or below, respectively, the absolute humidity H_i . The latter is already stored as a layer quantity and subject to advection. What remains is to calculate the saturation humidity for a cell.

First, for each layer $\mathcal{L}_i$ at altitude Θ_i , we compute the average atmospheric pressure using the barometric formula (BC09):

$$\Pi_i = \Pi_{sea} \cdot \exp\left[\frac{-g \cdot M_{air} \cdot T_{sea}}{R_0 \cdot T_0}\right], \tag{4.2}$$

where Π_{sea} and T_{sea} are pressure and temperature at sea-level, g is gravity, M_{air} is the molar mass of air, and R_0 is the Gas constant.

Then, for each grid cell p within each layer i we compute the saturated vapor pressure Π_i' using Teten's formula (the temperature is given in Celsius in this equation, while we use Kelvins elsewhere) (BC09):

$$\Pi_i'(p) = 0.61078 \exp\left(\frac{17.27 T_i(p)}{T_i(p) + 237.3}\right). \tag{4.3}$$

This equation holds for positive temperatures, and a variant exists for the negative case. In practice, we switch between them as appropriate. Finally, we are able to calculate the saturation humidity as (BC09):

$$H_i'(p) = \frac{M_{water}}{M_{air}} \cdot \frac{\Pi'(p)}{\Pi_i - \Pi'(p)}, \tag{4.4}$$

with M_{water} and M_{air} as the molar mass of water and dry air, respectively.

We then compare the saturated and absolute humidity and restore equilibrium by transferring $\Delta H_i = \gamma\,(H_i' - H_i)\Delta t$ from W_i to H_i at each time step according to a rate parameter γ (with $\gamma = 0.1s^{-1}$ in our simulation). This has the effect of decreasing water content and increasing absolute humidity during evaporation ($H_i' > H_i$), and vice versa for condensation ($H_i' < H_i$).

Inter-layer exchanges: Atmospheric layers that are entirely independent fail to capture uplift phenomena, such as convection, where relatively warm, moist parcels of air rise while cooler dryer parcels descend.

To account for such vertical transfers of temperature and moisture we incorporate a first exchange field, the *Convection factor*, into each atmospheric layer. This field depends on the interaction between the current layer $\mathcal{L}_i$ and the one above $\mathcal{L}_{i+1}$.

Convection is proportional to the disequilibrium in Moist Static Energy (MSE) between layers since it is this that causes air parcels to rise or fall: $C_i = \lambda(MSE_{i+1} - MSE_i)$, where λ is the convective coupling coefficient. The standard form for MSE is (BC09):

$$MSE_i = C_p \cdot T_i + g \cdot \Theta_i + L_v \cdot Q_i, \tag{4.5}$$

expressed in J/kg, and where L_v is the latent heat of vaporization, Q_i is the specific humidity of the air layer (and can be derived from H_i and Θ_i), and other quantities have already been defined. This serves to combine temperature, altitude and humidity into a single equation.

Vertical air motion can also be non-convective, solely and directly due to dynamic pressure differences between air layers. We express the local pressure difference as $U_i = \nu \cdot (P_{i+1} - P_i)$, where ν is the non-convective coupling coefficient. We then compute the total vertical velocity as $D_i = C_i + U_i$.

The vertical velocity field D_i is used to vertically transport T_i , H_i , and W_i upwards or downwards between layers towards a restoration of the MSE and pressure equilibria. To do so we start by computing the quantity to be exchanged between layers i and $i + 1$:

$$\Delta K_i = |K_{i+1} - K_i| \cdot \left(1 - \exp\left(-\frac{|D_i| \cdot \Delta t}{\Delta\Theta_i}\right)\right) \tag{4.6}$$

where K represents either T_i , H_i , or W_i , and $\Delta\Theta_i$ is the altitude difference between layers L_i and L_{i+1}. If D_i is positive, this quantity is transferred from K_i to K_{i+1}, and conversely if it is negative, with the proviso that neither K_i nor K_{i+1} can become negative.

Vertical displacements take place because the air itself moves. In order to account for such physical transport, we modify the corresponding pressure fields as follows:

$$P_i^+ = P_i^- + \mu D_i \tag{4.7}$$

where μ is the pressure coupling factor.

While this might seem non-physical, this solution is effective in practice. In particular, it is preferred to an advection scheme because of the extreme discretization of our problem: vertical transport occurs between layer pairs, which makes it ill suited to a continuous displacement approach.

Note that these inter-layer exchanges mean that each layer is no longer divergence-free. The risk here is that high vertical velocities may induce extreme pressure that would in turn tip the simulation into instability. While this might occur in theory, we have found that velocities remain bounded in practice.

Boundary conditions: The handling of boundary conditions in simulations is a perennial issue. We adopt the popular mechanism of wrapping the edges of our simulation to define a toroidal topology.

4.2.5 Terrain Encoding

In this section we consider the modifications to our layered simulation necessary to incorporate terrain-atmosphere interaction, including temperature and moisture transfers and the redirection of wind through orographic lift, expressed as a second inter-layer exchange term.

Orographic effect: Incorporating non-planar terrain topography first requires a choice in air-layer representation: should layer shape conform to the terrain or remain flat and fixed at a constant altitude? The first option could seem simpler as it obviates the need to process the intersection of atmosphere and terrain, but ends up significantly complicating advection because of the impact of within-layer variations in altitude on pressure.

Therefore, we choose the second option: we retain an atmospheric structure consisting of planar layers with constant altitude Θ_i , and set up specific boundary conditions and inter-layer interactions where the terrain impinges on a layer. To cope with this we borrow methods for obstacle handling from computational fluid dynamics and adapt them to account for airflow that veers either sideways (barrier jets) or upwards (orographic lift) depending on local topography.

To account for orographic lift we introduce the notion of orographic velocity damping $\sigma_i(p)$ based on how the terrain impinges on an air-layer $\mathcal{L}_i$ at position p. For a terrain with height $\Theta_{\mathcal{T}}(p)$ at p:

$$\sigma_i(p) = 1 - \min\left(1, \max\left(0, \frac{\Theta_{\mathcal{T}}(p) - \Theta_i}{\Delta\Theta_i}\right)\right). \tag{4.8}$$

This σ factor is used for damping the velocity field around obstacles. As depicted in Figure 4.4 it acts to deflect incident air upwards, producing non-convective vertical motion as explained in Section 4.2.4.2.

Heat balance: To account for heat fluxes in the terrain layer during the diurnal cycle and measure changes in heat and moisture due to interactions with the air, the following quantities are stored in terrain-specific fields: heat capacity $C_p(p)$ in Joules per kilogram per Kelvin, which is the amount of heat needed to raise up by 1K the temperature of one kilogram of a given material, determined by the land cover, be it open water, vegetation, grassland, or exposed ground; temperature $T_{\mathcal{T}}$ in Kelvin, indicating the current stored heat of the terrain cell, and water content $M_{\mathcal{T}}$ in g/m^2. The water content, in particular, is crucial to the cycle of evaporation and condensation.

We now turn to the process of heat absorption and release from the land surface. For terrain irra-

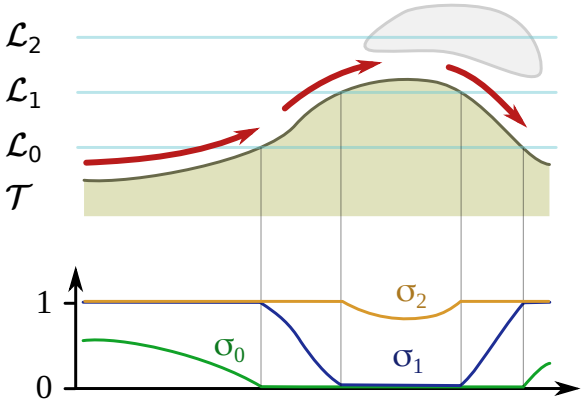

Figure 4.4: We use terrain topography $\mathcal{T}$ to compute the orographic velocity damping coefficients σ_i per layer $\mathcal{L}_i$. Damping velocity creates high and low pressure areas that transform into updraft and downdraft velocity components, which accounts for orographic effects, such as the lenticular cloud depicted here.

diation (and consequent heat absorption), the sun's position is tracked with a classical celestial model, and ray-marching used to compute the solar energy received by each parcel of terrain while accounting for self- and cloud- shadowing. Terrain temperature is updated according to the following irradiation equation (BC09):

$$\Delta T_{\mathcal{T}}(p)^{absorbed} = \frac{F_s \cdot (1 - B(p)) \cdot \cos(\theta) \cdot \Delta t}{D(\theta) \cdot C_p(p)}, \tag{4.9}$$

where $F_s = 340\ W/m^2$ is the sun constant, B is surface albedo (we use 0.15 for ground and 0.05 for water), θ is the incident angle of the sun, and $D(\theta)$ is the traversed atmospheric thickness from the law of cosines.

For simulating heat release into the atmosphere we employ a standard grey body model:

$$\Delta T_{\mathcal{T}}(p)^{emitted} = \epsilon \cdot \sigma_B \cdot \Delta t \cdot \frac{T_{\mathcal{T}}(p)^8}{C_p(p)}, \tag{4.10}$$

with $\epsilon = 0.9$ being surface emissivity and σ_B the Stefan-Boltzmann constant. Strictly speaking, ϵ should depend on terrain type, but, except for snow, these values do not vary significantly.

Atmospheric heat absorption: The final stage is to connect the ground with relevant atmospheric layers. We note $\ell(p) = \arg\min_{\Theta_i > \Theta_{\mathcal{T}}(P)} i$ as the lowest air layer that does not intersect the terrain at position p. There are then two categories of transfer:

1. Heat exchange: air temperature rises or falls towards a match with the heat released from the

ground. We model this as a standard conduction phenomenon:

$$T^{+}_{\ell(p)} = T^{-}_{\ell(p)} + \Delta T_{\mathcal{T}}(p)^{emitted} \cdot \kappa \tag{4.11}$$

where κ is the heat transfer coefficient proportional to air heat capacity and absorbance.

2. Water exchange: ground water evaporates into the air and atmospheric moisture condenses on the ground. This is exactly the same process as in intra-layer phase changes (see Section 4.2.4.2). When a phase change takes place, a quanta of energy is either released or absorbed affecting a change in the temperature of the ground, causing oceans to cool down due to evaporation. For a given transfer ΔH_i the corresponding change in temperature is:

$$\Delta T_i = \frac{\Delta H \cdot Q^{evap}}{C_p}, \tag{4.12}$$

where Q^{evap} is the latent heat of evaporation and C_p is the specific heat capacity of the terrain (water or ground).

4.2.6 Procedural Amplification

After the simulation process, we obtain a set of 2D grids containing different parameters (temperature, moisture, water content, velocity...), which is not sufficient to obtain a 3D volumetric rendering and know what type of cloud we should render.

To begin, a volumetric grid with regular voxel dimensions is derived by linearly interpolating corresponding elements of the water field between adjacent layers. However, it is important to capture local minima and maxima, which is why the number of layers and their specific altitudes are configurable during setup. In particular, in order to separate clouds into distinguishable bands it is necessary to add sandwiching layers of clear air.

We next detect what type of clouds we have to render in our simulation. For this purpose, we use a classification of cloud types: the standard taxonomy of major cloudforms (see Figure 4.5 and Table 4.3) is based on distinctions in appearance arising primarily from the altitude at which a cloud appears and turbulence caused by convectivity. This provides a convenient basis for classification because altitude Θ_i and convection C_i are readily available from our simulation.

One issue is that such hard classification boundaries can degrade spatial and temporal coherence. For example, there might be cells of one type scattered through a mass of a different type, or an advected cell may transition irregularly between types across frames. We solve this by grouping cells using k-means clustering of $5D$ cell vectors (incorporating position, time, convection, and altitude). This is seeded using

Type	Altitude (m)	Thickness (m)	Convection	Appearance
Stratus	$100 - 2000$	$100 - 1000$	low	pervasive and blanketing
Altostratus	$2000 - 6000$	$1000 - 5000$	low	sheet-like or wavy and possibly fragmented
Cirrostratus	> 6000	$1000 - 5000$	low	thin sheet, possibly with undulations
Nimbostratus	$100 - 7000$	$2000 - 4000$	low-med	thick, opaque, and featureless
Cirrus	> 6000	$1000 - 5000$	med	thin, wispy strands
Cumulus	$1000 - 2000$	$400 - 1500$	high	individual puffy clouds
Stratocumulus	$200 - 2000$	$200 - 400$	med-high	puffy and cohering
Altocumulus	$2000 - 6000$	$500 - 1000$	high	a puffy layer
Cirrocumulus	> 6000	$500 - 1000$	high	sheet with high-frequency structure
Cumulonimbus	> 200	$2000 - 12000$	very high	towering and lumpy

Table 4.3: Typical properties of different cloud forms.

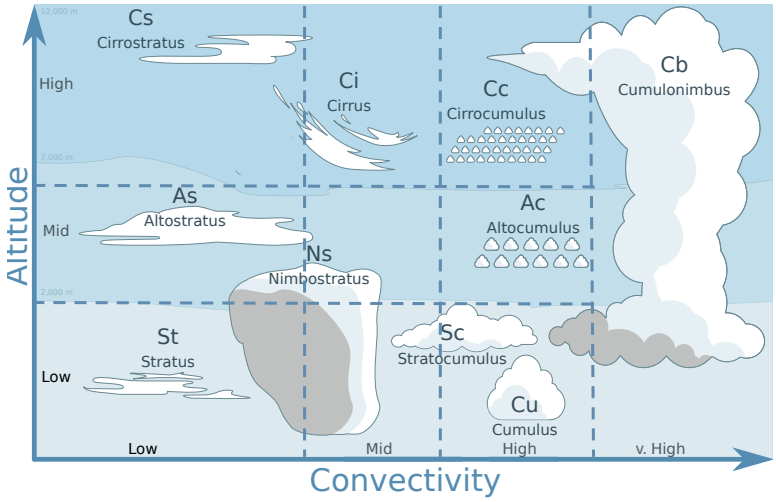

Figure 4.5: Our cloud classification scheme. The dashed lines indicate classification boundaries between cloudforms derived from Table 4.3.

the k-means++ algorithm (AV07) and the resulting cluster means (along with mean thickness) provide a consistent classification of the entire space-time cluster. Importantly, this is only applied in cases where a first classification pass reveals fragmentation since the process is computationally expensive.

We then apply different offsets and noise for the different types of clouds. In particular, stratiform clouds are centered at the altitude of the simulated layer (used as the isobar in our model); flat-bottomed cumuliform clouds are sampled above the layer, and high altitude clouds such as cirrostratus, which are naturally flat-topped, are sampled below the layer.

Since this spatial upsampling mechanism would only be sufficient for the most amorphous cloud structures, we also procedurally upsample cloud density using a time-varying $3D$ harmonic noise field $\mathcal{N}$ whose local spectral parameters depend on cloud type identified on a per-cell basis. For instance, fluffy cumuliform clouds require noise of higher amplitude and frequency than more wispy stratiform clouds.

Finally, if the animation is to be rendered at a finer time-step compared to the simulation, for instance to provide an animated background in a film or game, intermediate frames are also computed.

4.2.7 Results and discussion

We implemented our method in C++. Experiments were performed on a desktop computer equipped with an Intel® Xeon, clocked at 2.10 GHz with 31.3GB of RAM and 32 cores. The output of our system was exported Terragen4® for photorealistic rendering.

4.2.7.1 Authoring

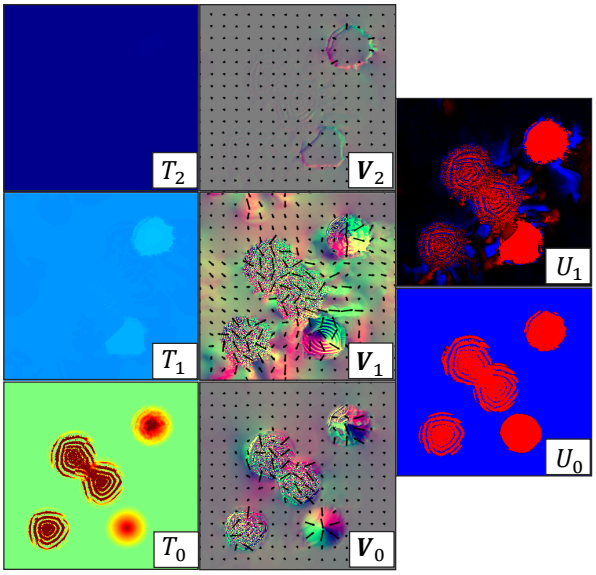

Figure 4.6: Convection cells form in a 3-layer simulation after the user paints hot-spots in the lowest temperature field. From left to right: temperature, 2D velocity visualized using both arrows and RGB colouring, and convection with a red (uplift) to blue (downdraft) heatmap. Successive layers appear from top to bottom with the convection field straddling layers.

Authoring simulation-driven content is known to be a challenge, primarily because controls tend to be indirect, such as setting and tuning initial conditions and simulation parameters. In addition to providing interactive visual feedback on the simulation (see Figure 4.6), which is essential to make indirect control possible, we assist users by providing three categories of authoring tools:

- **A direct painting interface.** The user is able to pause the simulation and paint directly into the temperature, humidity, water content and velocity fields of any layer, with immediate effect (as is done in Figure 4.6).

- **Scenario initialization.** This allows a user to configure the initial conditions on a per layer basis, including establishing a prevailing wind and seeding layer quantities either with noise or saved

Scene	Characteristics					Compute
	Layers	Altitudes (m)	Grid size	Duration (s)	Authoring	per frame (s)
Convection cells (Fig. 4.6)	3	10, 800, 1600	512^2	328	init, painting	2.73
Island convection (Fig. 4.7)	3	10, 800, 1600	512^2	600	init	3
Diurnal cycles (Fig. 4.8)	3	10, 800, 1600	256^2	3692	init	1.3
Orographic lift (Fig. 4.9)	3	10, 800, 1600	512^2	947	init	3.4
Mixed cloudforms (Fig. 4.10)	7	10, 2000, 4000, 5000, 7000, 8000, 9000	512^2	1350	init, timeline	6.75

Table 4.4: Main features -number of layers, layer heights, layer resolution, overall scene duration, and authoring mechanisms - and performance (average computation time per frame) for our validation scenarios.

Figure 4.7: Cumulus clouds forming above a pair of islands in a 3-layer simulation. An aerial view is shown inset in the bottom right of each image.

state from previous simulations.

- **Timeline events.** A scenario timeline can be constructed with wind and field events triggered along the domain boundary at specific times and for a set duration.

4.2.7.2 Validation

In this section we present a set of scenarios (summarised in Table 4.4) that validate our simulation against expected meteorological behaviour, including the formation of convective cells, changes due to diurnal cycles, and the impact of terrain. We conclude with a scenario that showcases the interaction of different cloud types. For all scenarios, we set the timestep to $\Delta t = 30s$, although anything in the range $10s$ to $240s$ works well for high-velocity scenes.

Our first and most straightforward experiment (see Figure 4.6) confirms that vertically-circulating convection cycles develop between layers ($n = 3$) in the presence of differential temperatures, as would be expected from uneven heat exchanges with the terrain. These are reminiscent of the Bernard cells that form in more constrained circumstances, such as a simmering liquid. Here, the user has directly painted hot-spots into the lowermost air-layer.

Our second scenario builds on the initial convection experiment to demonstrate the formation of cumulus clouds over an archipelago (see Figure 4.7) induced by heat transferred from exposed ground heated throughout the day. In this scenario air with entrained water vapour is fed landward, warmed by heat conducted from the sun-baked island surface and lifts, cools and condenses to form turbulent

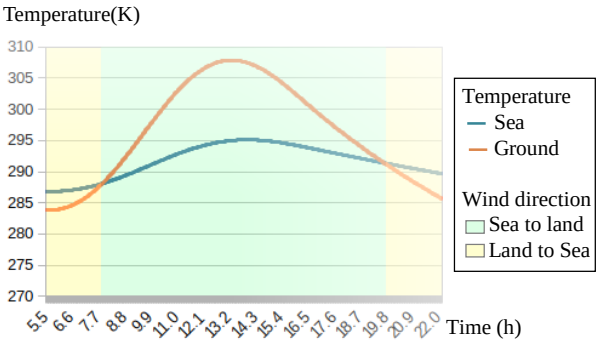

Figure 4.8: Diurnal cycles leading to a shift in wind direction at the interface between land and sea from onshore during the day to offshore at night.

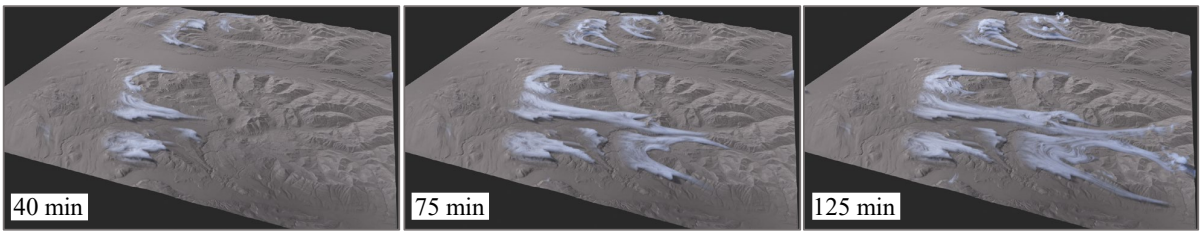

Figure 4.9: Orographic lift causes upslope cloud formation, showing one of the effects of terrain in our simulation framework.

cumulus clouds.

We use a diurnal cycle for a shoreline landscape to further test the influence of terrain type on irradiation and heat conduction (see Figure 4.8). The expected and evidenced behaviour is that a daytime onshore breeze will shift offshore during the night, due to different rates of absorption and release of heat between land and sea. A rendered image of the shoreline is shown in Figure 4.1.

To demonstrate the influence of topology on cloud formation we include an orographic lift scenario (see Figure 4.9) in which a moisture-laden wind is driven up a mountain slope to condense at cooler altitudes into stratus clouds. Such terrain interactions are a key contribution of our framework.

Finally, we showcase a combination of weather effects within a single scenario (see Figure 4.10). This encompasses the simultaneous evolution of different cloudforms within and between layers (l),

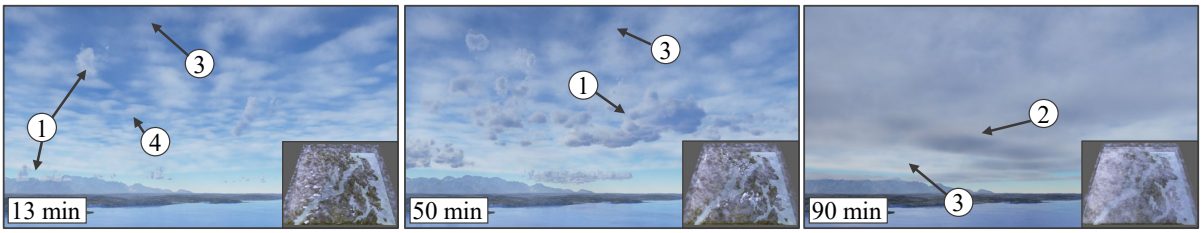

Figure 4.10: Multiple cloudforms within a single scene, including cumulus (1), altostratus (2), cirrus (3), and cirrostratus (4) clouds.

including cumulus ($l = [1, 3]$), altostratus ($l = 3$), cirrus ($l = 5$) and cirrostratus ($l = 7$). At higher altitudes we use inversion layers ($l = 3, 5$) to create separating bands of clear air.

Layers 3, 5, and 6 are initialized by the author with a prevailing wind direction and simplex noise in the H_i field. We also employ a timeline event to change the direction of the wind and introduce turbulence in layer 2 after a 30 minute interval.

4.2.7.3 Performance

Table 4.4 reports timings and statistics for the validation scenarios presented in this paper. The simulation performance of our CPU implementation is sufficient for authoring purposes at an average of 4.3s per frame for a 512^2 domain. While it is not possible to make definitive claims, we anticipate that this could be reduced to under a second for an optimized GPU implementation, given the known performance of multigrid methods on such an architecture (And18). In general, computation cost is roughly linear in the total number number of cells across all layers, as expected.

4.2.7.4 Limitations

Our hybrid approach has certain restrictions: some are inherent to a simulation strategy, but others could be solved through extensions to the framework.

The user is required to pre-set the layer parameters and finding an ideal configuration often requires trial-and-error adjustment. An auto-tuning framework, capable of automatically seeking and setting the optimal number and altitude of layers, would improve authoring. In addition, the planar resolution must be uniform and consistent across all layers. In theory, decoupling layer resolutions is possible, but would require changing the one-to-one mapping of updraft and convection between cells in vertically adjacent layers.

While we provide a variety of authoring tools, as with most simulations our controls cannot achieve the precision of procedural methods without degrading plausibility. For instance, a user can paint a shape directly into the water content field but this would force an unrealistic transition even if blended over several frames.

Currently, our skyscapes end abruptly at the 50×50km margin. In many cases this is not noticeable from a ground-based vantage. Nevertheless, it would be useful to couple our method with a synoptic-scale weather simulation (DYN06)) in order to provide coarser but still realistic boundary conditions for our simulation.

4.3 Coupling between a volcanic plume and clouds

4.3.1 Motivation

In this section, I propose an extension that combines volcanic plumes with the weather simulation method I just presented. Merging them leads to animations of new phenomena. In our work on plume and pyroclastic flows simulation in Chapter 3, we considered a theoretic atmosphere around the plume, with a constant temperature only depending on the height, and a linear wind. However, in the real phenomena, the surrounding air is heated by the very hot mixture of gas and ashes ejected from the vent, and the wind may change with time: real atmospheric conditions can influence the plume, as we can see for instance in Figure 4.12-right. Here, the wind has a different direction at different altitudes, as it pushes the plume and the ash rain in different directions. Furthermore, we did not consider the presence of water in the air that gets entrained by the column and is driven higher in the atmosphere where it condensates, forming clouds at the top and on the sides of the plume. These clouds are different from the other clouds animated in the first part of this chapter: they have a particular shape and appearance (see Figure 4.11-left), and they can quickly appear and disappear when they are located at the side of the plume.

We can observe some of these phenomena in Figure 4.11: in the left image, a cloud appears at the top of the plume because of air containing non-condensed water rising very quickly in the atmosphere with the high convection of the volcanic plume, cooling down rapidly, which causes clouds to appear around the column. Such cloud formation can also happen on the sides of the column (see Figure 4.11-right), involving the same phenomena.

The merging of the plume and weather systems also allows us to animate more phenomena around volcanic ejections, like ash rain (see Figure 4.12). Indeed, ash particles are heavier than the gas they are mixed with in the plume, and they fall, forming ash rain. Because of this sedimentation, the density of the plume decreases over time.

This work is still ongoing as we need to produce more results, work on rendering, and improving some aspects of the method towards publication, but the main technical parts are now stable. This work is a collaboration with Cilliers Pretorius and James Gain from Cape Town University, Guillaume Cordonnier from Inria Sophia Antipolis, and Jiong Chen from Ecole Polytechnique. The main contributions of this extension are the following:

- An interactive coupling method between a volcanic plume simulator and a weather simulator.
- An improved weather system handling rain and ash rain. I will only briefly mention these aspects since I did not work on this part myself.

Figure 4.11: Examples of clouds formation around a volcanic plume. Left: a small cloud forming on the top of the plume during an eruption (Sarychev Peak eruption on the island of Matua in 2009). Right: Clouds forming around the volcanic plume during the eruption of La Soufrière in 2021.

Figure 4.12: Examples of interaction between a volcanic plume and the clouds/the weather around. Left: Ash rain falling down from a plume (Eruption of Tavurvur, New Guinea, in 2014). Right: A plume under a wind during the eruption of Shinmoedake in 2011. Ash is falling and pushed by the wind in a different direction than the plume.

4.3.2 Study of interaction between volcanic plumes and weather phenomena in geoscience

As in my other projects, I discussed with a volcanologist to better understand the underlying phenomena and their visual effects. Geoscientists have indeed worked for many years on these topics. Glaze et al. (GBW97) studied the transport of atmospheric water vapor through a volcanic column, causing condensation and cloud formation at higher altitudes. Gilbert et al. (GL94) observed the origin of the formation of accretionary lapilli, which is how ejected ash or other particles aggregate to form bigger particles (lapilli) that fall down. Van Eaton et al. (VEMH+15) studied the role of hail formation in this phenomenon, and it triggers rapid ash aggregation. Finally, several articles studied lightning formation in volcanic columns (JWL+08; SGVE+21) and Cimarelli et al. (CAIK+14) reproduced this lightning

formation in indoor experiments.

4.3.3 Coupling of the plume and the atmosphere

Since the velocities involved in the volcanic plume are much higher than for the weather simulation, the time steps for the plume need to be smaller. We use a time step of 0.02s for the plume and 1s for the weather. This means that several steps of the plume simulation happen between two steps of the weather simulation.

Data transfer: First, coupling the two systems implies transferring data between them. The data transfer from the plume to the atmosphere happens just before the weather step, and the data transfer from the atmosphere to the plume happens just after the weather step. We transfer several parameters:

- The temperature, in both directions, so that the plume takes the real temperature of the surrounding air; and the plume temperature acts like a local constraint for temperatures in the weather system, and have an influence on cloud formation.
- The moisture, in both directions, in order to track the vertical moisture displacements due to the plume that lead to cloud formation.
- The wind velocities, only from the weather system to the plume system, so that the plume considers the real wind around it.

To transfer data from the weather system to the plume system, we consider the region in the air layers close to the plume, so that it corresponds more or less to the air entrained by the plume. For each air layer, we compute a mean value for the different parameters to be transferred in the cells in an annulus around the plume with an internal radius equal to the radius of the plume r_{plume} and an external radius of $3\,r_{plume}$, as shown in Figure 4.13. Using linear interpolation, we deduce values for temperature and wind at every altitude between the air layers to compute the behaviour of the plume.

To transfer data from the plume to the weather system, for each air layer, we consider the closest plume slice - if it is within the altitude range of the air layer. The temperature (respectively the moisture) of the cells overlapping the plume area takes the value of a linear function reaching its maximum at the center of the plume and its minimum on the borders with a mean value equal to the temperature (respectively the moisture) value of the closest slice.

Ash rain: To animate ash rain, we need to get a visual consistency in the final rendering: the ash has to fall from the plume spheres, that are the rendered part of the volcanic plume. We use the weather system to simulate ash rain the same way rain is simulated, and to get this visual consistency we transfer

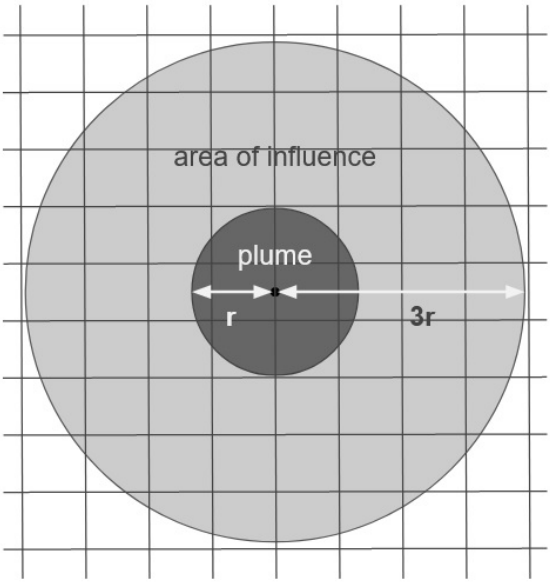

Figure 4.13: Region covered by the plume and its area of influence in the grid. We transfer data from the atmosphere to the plume system by computing the mean value in all cells in light grey. We transfer data from the plume to the weather system by enforcing values in the cells in dark grey.

ash loss data from the plume spheres to the weather system. Between every weather time step, every plume sphere stores the ash loss density information, and this ash density is transferred to the weather system just before the weather time step. We add these densities in the cells of the closest air layer to the spheres, which leads to ash rain in the weather system.

4.3.4 Results

4.3.4.1 Validation of plume dynamics

To check that the data transfer works and that the plume still raises correctly when taking the parameters of the surrounding air from the weather simulation, we produce the same behaviour curves as we did in the previous chapter and we compare them with the curves we obtained with a theoretic atmosphere in Chapter 3. We can see in Figure 4.14 the curves for density and speed depending on altitude and the trajectory of the plume under a wind with a magnitude increasing linearly with altitude and reaching 45m/s have the same shape as what we obtain on Figure 3.5 in the previous chapter with a theoretic atmosphere. We set a higher initial speed of the plume (170m/s instead of 150m/s) because we changed the altitude of the vent to 2500m, and because of the heating of the surrounding air by the plume.

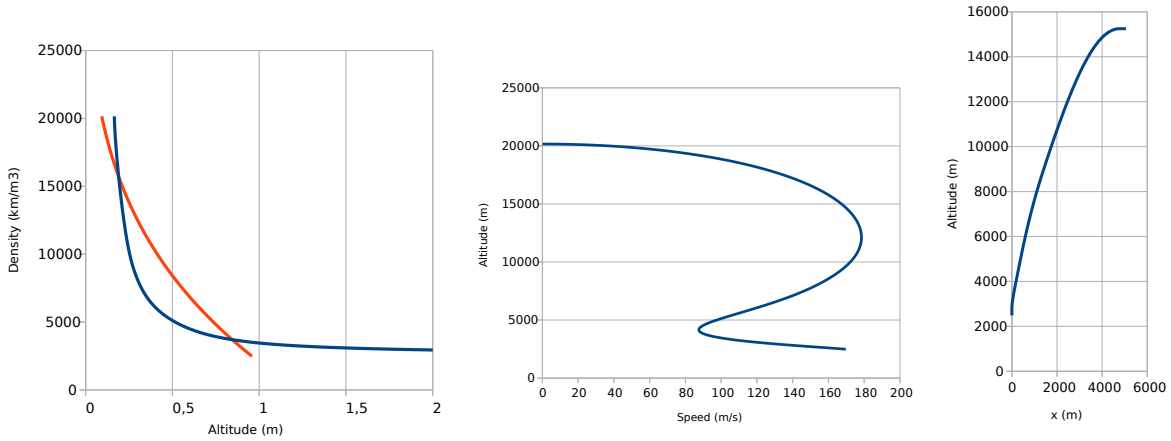

Figure 4.14: Left: Comparison between the density of a plume slice (blue curve) and the atmosphere (red curve) with altitude. Middle: Evolution of the vertical speed of the plume with altitude, with an initial speed of 170 m/s. Right: Trajectory of a plume under a wind originally set linear with a maximum of 50m/s at high altitude, and changing with time in the weather simulation.

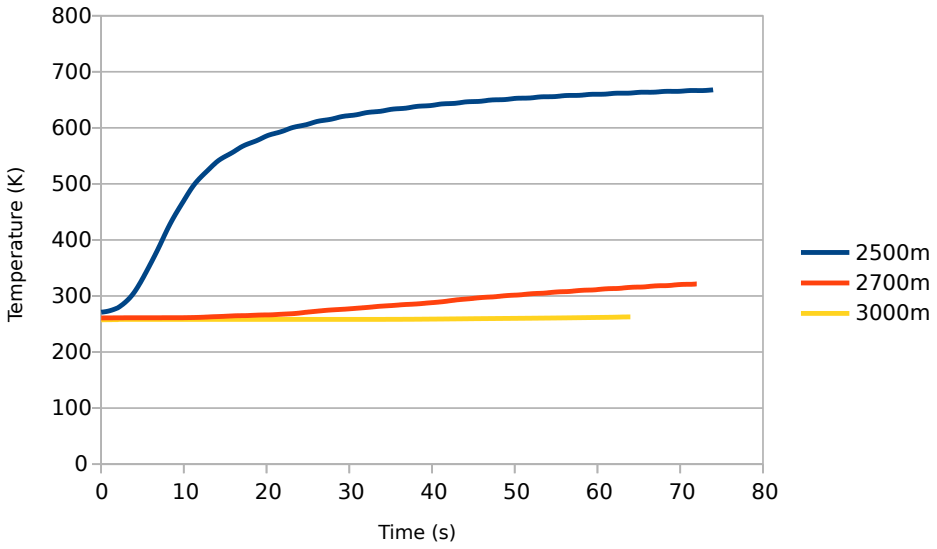

Figure 4.15: Smoothed curves showing the evolution of the temperature around the plume over time at altitudes of 2500m (where the vent is), 2700m, and 3000m.

4.3.4.2 Coupling of the plume and the atmosphere

To validate our approach, we tested the effect of the plume on the nearby atmosphere. The air around the plume in the weather system heats up as it should with the diffusion of high temperatures around the vent. This effect is at its maximum at the vent altitude and decreases with altitude as the plume itself cools down. Figure 4.15 shows the evolution of the temperature in the area of influence of the plume (as defined in Figure 4.13), used in the plume simulation instead of the theoretic atmosphere temperature. The curves are smoothed out and show temperatures at the altitude of the vent, 200m above and 500m above the vent. At the altitude of the vent, the temperature raises a lot before stabilizing; 200m above

the vent, the temperature raises only by a few dozens Kelvin; and 500m above the vent, the temperature almost stays the same.

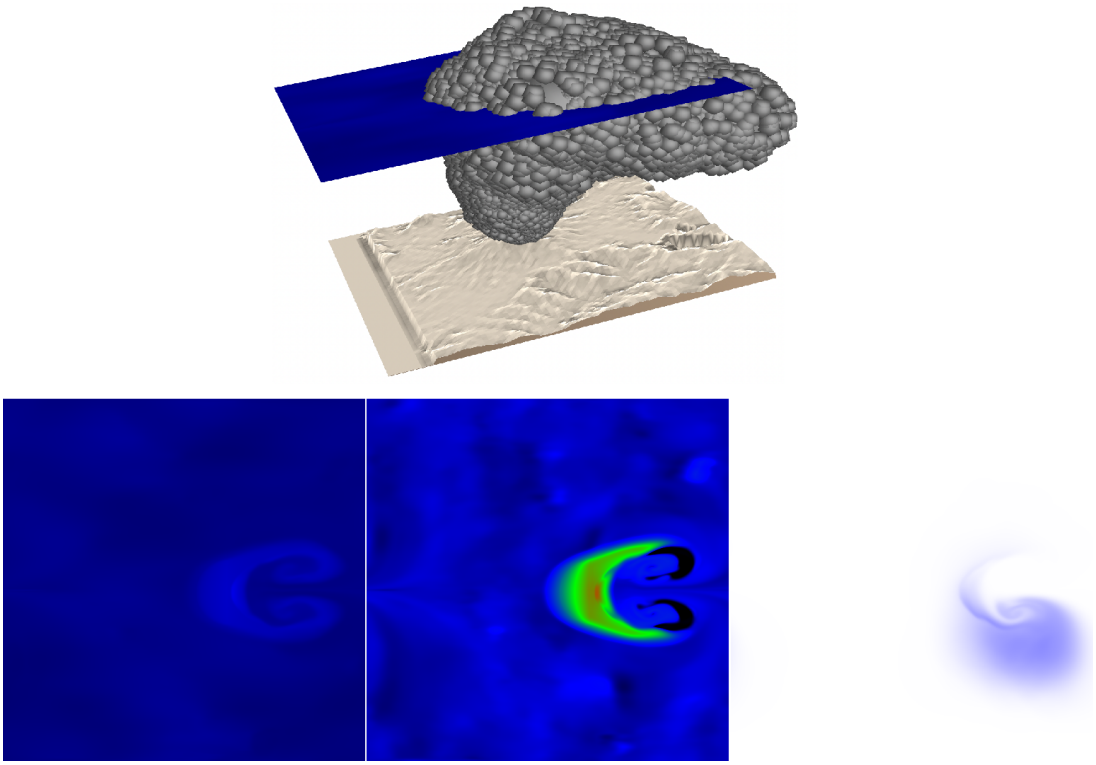

Figure 4.16: Top: A volcanic plume encountering an air layer at 16500m of altitude. Bottom: Corresponding maps of 30km by 30km showing condensed water (left), temperature (middle) and ash content (right) at 16500m after 250s of simulation.

The effect on temperature, condensed water and ash content near the plume can also be seen in 2D maps of the air layers in Figure 4.16. The bottom row show maps for these parameters at an altitude of 16500m after 250s, for an air layer reached by the volcanic plume as shown on the top. Where the plume is located, we can see a raise in temperature, condensed water and ash content values, that become advected by the local wind.

4.3.4.3 Future work

As mentioned before, this extension is still an ongoing work. We still need to improve our system and produce results for different test cases.

1. The first test case should be able to show the formation of clouds above a volcanic plume and on its sides. For this test case, we need to improve the moisture transport by the plume to simulate the cloud formation; to set up the rendering parameters for this type of clouds to make it look like the clouds in Figure 4.11; and to produce a complete simulation and rendering of this phenomenon. In this test case, we will also initialize a layer of clouds to see a small hole forming in it.

2. The second test case would consist in setting up a volcanic plume under a wind with different directions at different altitudes, so that the plume bends under it, ash falls as a rain and gets pushed in another direction like in Figure 4.12-right. For this test case to work, we need to finish setting up the ash transfer to the weather system to make it fall like rain; to set up the rendering for the ash rain; and finally to render a simulation for showcasing ash rain.

4.4 Conclusion

In this work, we have shown that plausible and diverse animated wind and clouds can be generated at interactive rates, with or without the effect of a volcanic eruption, and then further enriched with consistent procedural details at the rendering stage. Our layered physically-based model provides a unique way to efficiently simulate the formation and evolution of wind and clouds over time, thanks to its ability to balance sparse vertical detail with a higher level of horizontal detail, enabling capture of the most relevant meteorological effects. This simulation allows the user to interactively launch and tune weather scenarios, before the offline rendering of an animated skyscape. The coupling with the volcanic plume system enables more diversity in the cloud types and the possible scenarios, with the addition of ash rain for instance.

Our method takes the presence of water bodies and the topology of the terrain into account. Moreover, our taxonomy for cloud types enable us to compute consistent up-sampling of our simulation data, leading to the visual complexity required by games and feature films.

A first extension to tackle in future work would be the inclusion of more phenomena, like thunder coming from clouds or plumes, hail, snow, and other cloud sub-types. This could be done using stochastic, procedural models controlled via probability heatmaps generated through simulation. Enabling the inverse modeling of precise events (e.g., “rain begins at 8 am") would also ease authoring of complex and precise scenarios. Modeling the long term effects of volcanic eruptions on the global weather with sulfur dioxide would also be a nice addition, leading into cooling and red skies.

Lastly, our model could be coupled with other natural phenomena frameworks to improve consistency in virtual landscape modelling: run-off and erosion simulation (CGG+17) as well as vegetation simulation (GLCC17) come immediately to mind, as these phenomena are heavily dependant on weather and ash deposit coming from volcanic plumes.

Chapter 5

Surface animation for lava flows

5.1 Introduction

Volcanoes are always caused by magma coming up to the surface of the Earth, but magma does not always come out as lava flows as those shown in movies. Lava is a complex non-Newtonian fluid - i.e. its viscosity depends on stress. It can show many different behaviors depending on its composition. Notably, silicate lava is not fluid enough to form lava flows and stays around the vent, thus blocking the path for more volcanic material to come out until the pressure is too high, and an explosive volcanic eruption happens. Non-silicate lava is more fluid and can form flows; but again, we can differentiate several types of lava flows.

Figure 5.1: Left: a'a lava flow. Right: pahoehoe lava flow.

Two main categories of lava flows can be distinguished: pahoehoe and a'a flows, where a'a are more viscous than pahoehoe. Both can happen in the same eruption if the viscosity of the lava changes. A'a flows look more chunky with more non-uniform solid lava rocks at the surface (see 5.1-left), while pahoehoe flows have a smoother surface (see 5.1-right).

Previous work in computer graphics already tried to mimic the behavior of both a'a (SAC+99) and

pahoehoe (SSJ+14) flows. However, some interesting phenomena happen at the surface of pahoehoe flows and have never been represented before. Therefore, we focus on this specific type of lava flow in this chapter.

Pahoehoe flows mostly happen in Hawaii, where the most active volcano is the Kilauea. As lava is quite fluid, lava flows can travel a long distance before hardening completely. This is why the volcanoes in Hawaii are rather flat.

Figure 5.2: Left: pahoehoe lava field ©Sean Goebel. Middle: pahoehoe lava folds ©Bryan Lowry/lavapix.com. Right: a new flow of pahoehoe lava forming out of the field ©Bryan Lowry/lavapix.com.

Although Pahoehoe lava flows exhibit 3D behavior, the apparent surface motion can be represented by a 2D velocity field, causing local compression and stretching phenomena. When rigid crust forms at the surface of the lava flow, it moves along this 2D velocity field.

When the lava comes out of the vent, it first has a yellow/orange color due to its temperature. If the moving lava is thick enough, a crust slowly starts forming at the surface while it cools down with the effect of radiation. The thickness of this crust increases where the 2D surface velocity field results in a compression, and decreases in case of stretching. The behavior of this solid crust at the surface of a liquid flow can be compared to a cloth: the crust is driven by the underneath flow, and folds where the compression is to high, e.g. when the flow gets blocked by obstacles (see 5.2-middle or this video).

To understand the behavior of pahoehoe flows, the whole lava field has to be considered in its entirety: indeed, even if a crust is formed at the surface and the flow does not seem to move, liquid lava continues to run underneath (see 5.2-left or this video). The various apparently solidified flows in the lava field slowly swell as the lava accumulates. When the pressure of the lava inside the crust is too high, the crust tears and a new flow emerges (see 5.2-right).

In this chapter, we focus on the animation of surface effects of pahoehoe flows, and introduce a new multi-layer model to represent them. We use a pre-existing Eulerian simulation method for the flow, enabling us to focus on crust thickness computation, and on the possible apparition of folds at the surface. In this case, we add geometric crust folds that are both advected along the velocity of the lava flow and may collide with other folds. This may retroactively change the local surface velocity, and crust thickness of the flow. We visualize this effect using a combination of implicit surfaces and temporally consistent textures.

5.2 Related work

5.2.1 Lava flows models in Computer Graphics

Different previous works aiming to animate or texture lava flows have already been presented in Chapter 2. These paper are summed up in Table 5.1, with the categories of model they used for either the flow or the lava crust, or both.

	Simulation	Textures
(SAC+99)	Yes (SPH)	Yes (procedural 3D texture)
(NKL+07)	No	Yes (patch-based texture)
(CBL+09)	Yes (SPH)	No
(SSJ+14)	Yes (MPM)	No
(ZKL+17)	Yes (SPH)	No
(GGV+19)	No	Yes (texture mapping)

Table 5.1: Previous work in lava flow animation and/or texturing

I also want to focus on a real-time lava flow demonstrator coded by Chahi and Sahy (CS15) in collaboration with the volcano museum *La Cité du Volcan* in La Réunion, where a famous active volcano, the Piton de la Fournaise, is located. They implemented an interactive visual simulator for lava-flows, to be displayed in the museum and used by the visitors. The latter can interactively place lava sources on the terrain and observe a lava flow animation in real-time in VR. They used a GPU implementation of the work of Mei et al. (MDH07) to simulate the flow and allowed the lava to harden, thus becoming part of the ground. They textured the flow with dynamically advected textures inspired by Neyret (Ney03). Figure 5.3 shows some of their results, and other images and videos of their animations can be found on their website. They kindly allowed us to use their lava simulation as a base for the more accurate lava crust model introduced in this chapter.

Figure 5.3: Some results of lava flows animation in the Volcano Simulator (CS15), computed in real time. Note that the crust being an advected texture, it cannot thicken and fold.

Note that other fields of research like numerical analysis also discussed lava flow models (BSS14).

5.2.2 Taking inspiration from lava flow studies in geoscience

Lava is a non-Newtonian, incompressible fluid that can show many different behaviours since its viscosity depends on temperature. While non-Newtonian fluids were extensively studied in geoscience, lava in itself not that much, because of the difficult conditions of observations such as very hot temperatures and the risk of walking on unstable ground with lava underneath. Little theory has been produced, the work in this field being principally based on observations.

However, some papers address the formation of crust, like Wylie et al. (WL98) who analyzed the stability of the cooling surface under the influence of a straining flow; or Peterson et al.(PT80) observed the transition of lava transitioning from pahoehoe to a'a in an eruption on the Kilauea volcano in Hawaii, which happens in certain cases when the cooled surface forms bigger chunks of rock.

Studying other types of volcanic eruptions, like explosive eruptions, can also help in understanding some aspects of lava flows when literature becomes insufficient. For instance, Kaminski et al. (KJ97) study the formation of pumice (which are low-density vesicular volcanic rocks) in Plinian eruptions. This is quite similar to the dilatation of lava in contact with cold air and can help with the comprehension of the cooling of lava flows.

The study of other viscous materials can also be generalized to lava flows: Ribe et al. (Rib04) describe the coiling of viscous jets like honey or toothpaste, but this can also describe how the lava flow with a cooled crust at its surface begins to form wrinkles when it hits an obstacle.

Finally, in our work, we took the advice of a volcanologist to propose a simplified model of the crust formation phenomena. Previous work in volcanology will therefore allow us to justify some of our simplifications, similarly to the approach in Chapter 3 where we reduced the problem of volcanic plume formation to a single dimension. More precisely, a compressible lava flow (which happens if there are a lot of bubbles in the lava) can be modeled by treating several problems separately, namely the internal pressure, the velocity field, and the flow thickness, which is a simplification (Jau91). We propose a similar simplification in our work by treating the temperature field, the velocity field, and the flow thickness separately when we compute the crust thickness.

5.3 Overview

The model I present in this chapter is again a multi-layer model with three layers, each one representing one level of the lava flow simulation.

- First, an Eulerian simulation is used, where each cell contains a certain lava height. Knowing the height and temperature of lava and ground in each cell and its neighbours, a flow can be

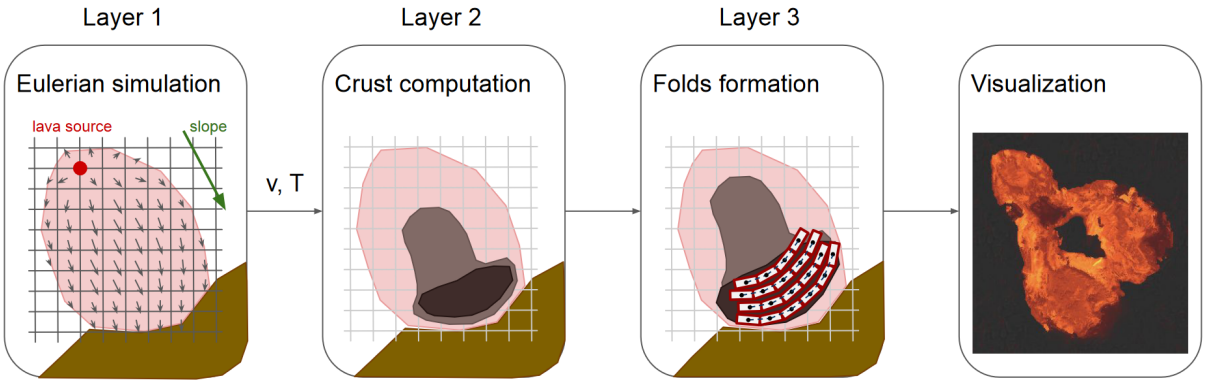

Figure 5.4: Overview: we take the output of an Eulerian simulation of a lava flow for the computation of the crust thickness, and we create fold structures where the crust is compressed.

computed, and we can obtain the new lava heights in each cell at the next time step. A velocity field is deduced from this flow. The implementation of this part relies on the existing code from Chahi and Sahy (CS15). We then use the velocity and temperature fields it outputs.

- Then, in a second layer, we take the velocity field from the Eulerian simulation and compute several parameters in each cell of the grid: the surface velocity of the flow, and the crust thickness. We also detect compressed regions in the flow, enabling us to deduce where folds can appear. I cover this sub-model in Section 5.4.

- Finally, in a third procedural geometric layer, we create folds at the surface of lava wherever possible. We also define how these folds move and collide. We detail this layer in Section 5.5.

Once these three simulation layers are used, we still need to visualize the results. Our interactive simulation system allows to watch the flow, observe the appearance of folds, and enables us to add new lava sources anytime. We compute a mesh around the flow and especially the folds, using an implicit deformer to adjust the height of the mesh. Finally, we export our simulation data (crust thickness, the surface velocity field) so that they can be used to generate time-consistent textures which will be mapped on the mesh. I go over all of these in Section 5.6.

Please note that this chapter is still an ongoing work. We will present the main model for the crust generation and animation, but results are not final and would need further refinement.

5.4 Surface velocity and crust thickness

The surface simulation of the lava flow uses an Eulerian simulation as a base (CS15; MDH07). In this simulation of lava, each cell of the grid has a certain height of lava in it. At each time step, lava is transferred from one cell to its neighbours, knowing the height of fluid in each cell and other parameters like

Parameter	Notation	Value
Lava density	ρ	$2.7\,10^3\,kg.m^{-3}$
Thermal conductivity	k	$2.7\,W.m^{-1}.K^{-1}$
Heat capacity	C_p	$10^3\,J.kg^{-1}.K^{-1}$
Lava kappa	κ	$\frac{k}{\rho\,C_p} = 10^{-6}\,m^2.s^{-1}$
Air temperature	T_e	$300K$
Initial temperature of lava (liquidus)	T_l	$1473\,K$
Solidification temperature of lava (solidus)	T_s	$1273\,K$
Stefan-Boltzmann constant	σ	$5.67\,10^{-8}\,W.m^{-2}.K^{-4}$
Spring constant for forces between particles	k_{spring}	$10^3\,N.m^{-1}$
Particle length	L	$0.5\,m$
Time step	dt	$0.02\,s$
Cell size	d_{cell}	$0.5\,m$

Table 5.2: Constants used in our simulation

Parameter	Notation	Unit
Time of air contact	t	s
Lava temperature	T	K
Flow velocity	$\vec{v}_{flow}$	$m.s^{-1}$
Surface velocity	$\vec{v}_{surf}$	$m.s^{-1}$
Lava thickness	h_{lava}	m
Crust thickness	h_{crust}	m
Fold thickness	ℓ	m
Particle orientation	$\vec{d}$	/
Lava viscosity	ν_{lava}	$Pa.s$
Crust viscosity	ν_{crust}	$Pa.s$

Table 5.3: Parameters used in our simulation

the temperature. A velocity field is deduced from these transfers of lava between cells. This simulation outputs grids of values for velocity, temperature, and lava height. In this section, I will explain how we compute the surface parameters (surface velocity, crust thickness) from this data.

All the notations and parameters used are summed up in Tables 5.2 and 5.3.

We distinguish several velocity fields for our lava animation:

- The real 3D velocity field of a lava flow; we do not actually use this one in our simulation;
- The 2D velocity field of the lava flow $\vec{v}_{flow}$ that the simulation described above outputs. This field corresponds to the mean of the horizontal projection of the 3D velocity field of the flow;
- The surface velocity field $\vec{v}_{surf}$: it stand for the velocity field at the surface of the liquid part of the flow, which guides the crust of the flow.

Figure 5.5 depicts a side cut of a lava flow to help understanding the main notions we use, namely the lava thickness h_{lava}, the crust thickness h_{crust} and the fold thickness ℓ.

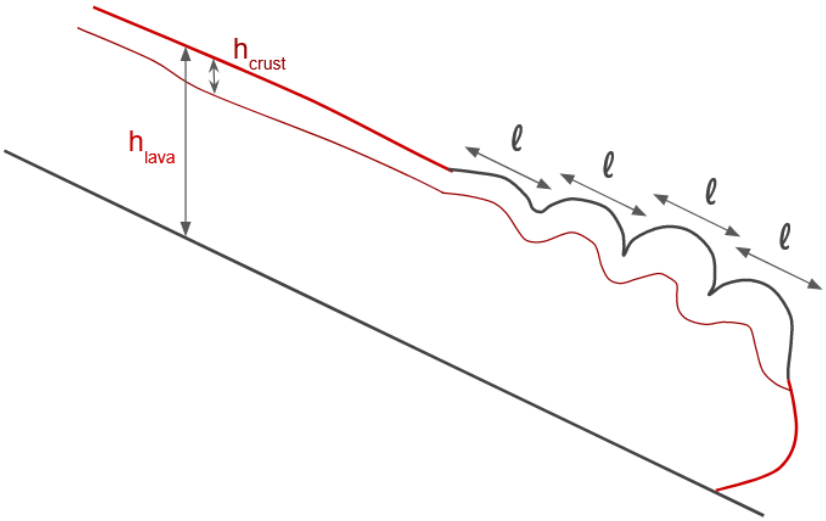

Figure 5.5: Side cut of a lava flow with a crust and folds.

5.4.1 Lava flow analysis

As we want to animate the surface of the flow, we need to deduce a surface velocity $\vec{v}_{surf}$ from the flow velocity $\vec{v}_{flow}$ given by the Eulerian simulation. These two velocities are initially the same as the flow is very thin, with no crust. When a crust appears, it cannot behave like the liquid lava underneath since the crust is more rigid and constrains the liquid lava at the interface. To model this, we apply some smoothing on the velocity, depending on the crust thickness, described next.

5.4.2 Crust thickness model

The evolution of crust thickness at the surface of a lava flow depends on two phenomena: radiation and divergence of the surface velocity field of the lava in contact with the crust.

Radiation is the emission of energy from the lava surface into the air in contact with the flow, and it causes the cooling of the surface with time.

We directly compute the crust thickness knowing the time spent in contact with the air, using the following equation derived from the combination of the heat equation and a radiation constraint at the interface between the flow and the air:

$$h_{crust}(t) \approx 2\sqrt{\kappa\, t} \tag{5.1}$$

$$\frac{1}{h}\frac{\partial h_{crust}}{\partial t} = \frac{1}{t} \tag{5.2}$$

where h_{crust} is the crust thickness, t the duration the lava has been in contact with the air, and $\kappa = \frac{k}{\rho\, C_p}$,

with k the thermal conductivity of lava, ρ its density, and C_p its heat capacity.

Divergence: The second phenomenon involved in the formation of a crust at the surface of the lava flow is the divergence of the surface velocity field. If the crust is locally compressed, its thickness increases; if it is stretched, its thickness decreases. This can be written as:

$$\frac{1}{h_{crust}} \frac{\partial h_{crust}}{\partial t} = -\nabla \vec{v}_{surf} \tag{5.3}$$

where $\vec{v}_{surf}$ is the velocity field at the surface. We don't always apply this equation to compute the crust thickness, but only once the thickness is already high enough (>1cm).

Furthermore, if the crust thickness decreases too much due to the stretching, the contact between the flow and the air changes, and we reset the time parameter t so that radiation can correctly be taken into account again, using $t = \frac{h_{crust}^2}{4\,\kappa}$.

Finally, if the lava flow is too thin and in motion, a crust cannot form: we set to zero the crust thickness where the lava thickness is smaller than 20cm and has a non-null velocity. With all of this, we can compute the crust thickness at the surface of lava everywhere at any time.

To obtain more accurate measurements of the crust thickness, we use Lagrangian particle samples that are advected along the surface velocity field, and are used for the computation of crust thickness in the grid. To do that, we track the time since the emission of these particles, and use it to compute the crust thickness where each one of them is, and finally compute a mean in each cell of the grid.

5.4.3 Folding of the lava crust

Once we know the travelling speed and thickness of the crust everywhere on the lava flow, we determine where to form folds.

Several requirements have to be fulfilled to begin the creation of a first fold:

- The crust has to be thick enough. We take 1cm as the minimum crust thickness enabling folds to form.

- The lava flow also has to be thick enough. We set this minimum thickness to 50cm.

- The compression applied on the crust has to be high enough to induce folding. For instance, this happens when the flow hits an obstacle. To detect that, we compute the maximum lava height difference between one cell and its neighbours. This height difference is maximal on the borders of the flow, i.e. between cells with and without lava in them; and it is even higher on a border

where the lava hits an obstacle and accumulates. We set a minimum height difference at 20cm so that it only detects the borders where lava accumulates.

For the following folds, we use less strict conditions: we only require the crust to be thick enough (>1cm), as well as the lava flow (>30cm). We do not need to set a criteria involving compression, since the next folds appear naturally behind the previous ones.

5.5 Fold structures

In this section, I will explain how we create and animate folds like the ones on Figure 5.2-middle, at the surface of a pahoehoe lava flow. These folds appear when the surface crust is compressed in one direction, orthogonal to the surface velocity field.

The general idea of our method to represent folds is to create and chain particles at the surface of the flow, then advected by surface velocity. The next section first details the parameters of the oriented particle model we use, before detailing the fold formation algorithm.

5.5.1 Fold formation

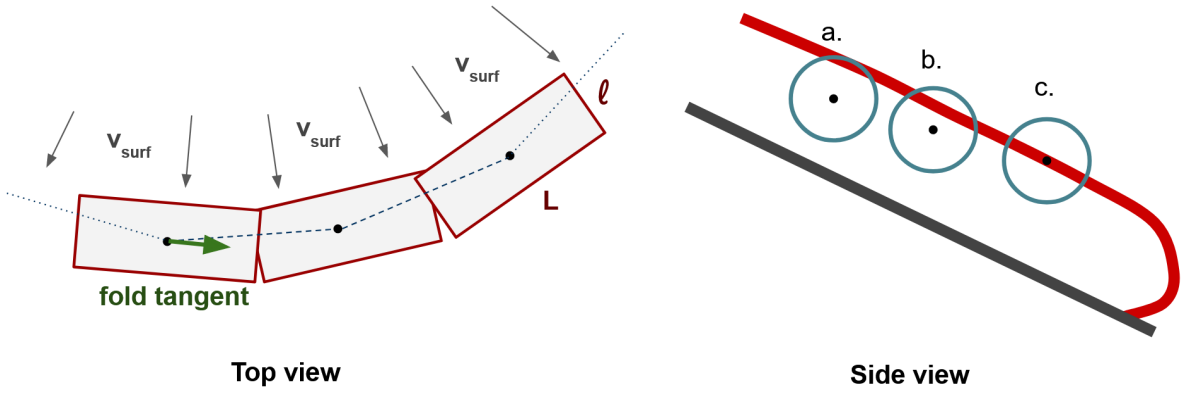

Figure 5.6: Left: Basic fold structure (top view). A fold is made of a chain of oriented particles physically represented by cylinders. L is the length of the cylinder and the distance between particles, and ℓ its diameter. Right: Offset of a fold (side view). a. When the fold is created, it is first invisible under the surface, and the offset is set at $-\frac{\ell}{2}$. b. The offset increases over time. c. The offset eventually reaches zero, placing the fold at the surface of the flow.

The particles. We use oriented particles, visualized as cylinders, to model the progressive formation of folds, where the cylinder's radius is related to the thickness of the future fold. Several parameters define these cylinders, and by extension the associated particles:

- The fold tangent $\vec{d}$, which is initialized as the normalized horizontal vector orthogonal to the surface velocity field. When the particle is chained to other ones, the tangent is the normalized vector between its neighbours;

- The length L, which is the height of the cylinder. It corresponds to the distance between two particles in a chain, and we preset it as a constant in Table 5.2. If this constant is too large, the fold will not be able to bend properly and will lack details; but too many particles in a fold are not necessary so the length should not be too small either.

- The fold thickness ℓ, depicted in Figure 5.5, and which corresponds to the local wavelength of the folds. This is the diameter we set for the cylinder. It depends on the lava and crust thicknesses and viscosities, and is computed at the formation of the fold.

- The vertical offset of the particle relative to the surface of the flow. It is initialized at $-\frac{\ell}{2}$, to make the particle invisible under the surface, and slowly increases up to zero, placing the particle at the surface of the flow (see Figure 5.6-right). This way, the folds do not immediately appear at the surface with their final height: they are first quite flat, and then they gain in visible amplitude. This approach was inspired by a former model for cloth wrinkles (RPC+10). This offset is not the same everywhere within a fold: it reaches its maximum in the main part of the fold in the middle but decreases on the sides to ensure a smooth transition with the surroundings.

Fold initialization. To create a fold, we first detect a cell where a fold can appear to initialize it with a first particle, and we then chain particles in both directions along the fold tangent vector. When we detect such a cell in the grid, we create a first particle at the center of the cell. The cylinder attached to the particle is oriented orthogonally with respect to the local surface velocity. We set its thickness to:

$$\ell = \sqrt{h_{crust}\, h_{lava}} \left(\frac{\nu_{crust}}{\nu_{lava}} \right)^{1/6} \tag{5.4}$$

where the ν parameters denote the respective viscosities and h the respective thicknesses of liquid lava and crust. This is a general formula for computing the wavelength of a fluid or semi-fluid material folding on top of another one, also used for instance by Cordonnier et al. (CCB+18) to compute the wavelength of folding earth crust layers.

Fold creation. To create a fold starting from one particle P, we iteratively add particles on both sides until no more particles can be added. For better accuracy, the computation of the position of the next particle P_{next} of the chain is done in two steps, as shown on Figure 5.7-left. On this image, p is the

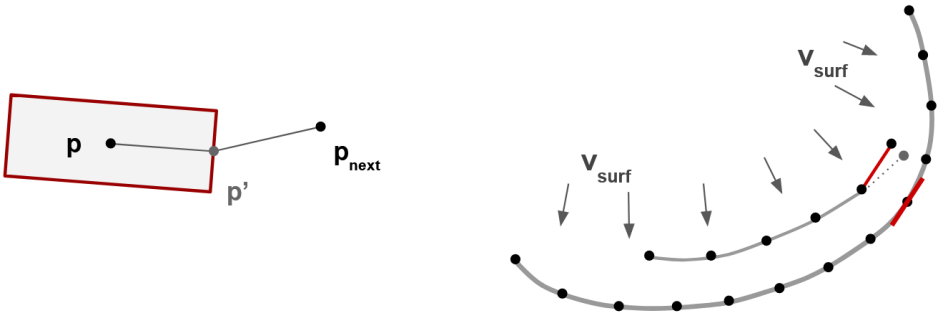

Figure 5.7: Iterative formation of a fold, particle after particle. Left: creation of the next particle in a fold in two steps (top view). Right: if this new particle is too close to a neighbouring fold, the fold tangent of the closest particle is used instead (top view).

position of the existing particle, $p' = p + \frac{L}{2}\,\vec{d}$ an intermediate position and d' its tangent, and $p_{next} = p' + \frac{L}{2}\,\vec{d'}$ the position of the next particle.

The new particle can only be added if there is crust on the lava flow at its position. Therefore, we check the presence of a crust that is thick enough (>8mm), and that the whole region covered by the cylinder associated to the particle is on the lava flow. If one of these criteria is not fulfilled, the new particle is not added and the chain stops there.

The newly formed fold can also be extended later with the same iterative method: we regularly check if it is possible, which can happen if more crust is formed on the side.

Following folds. Once the first fold is created, the following folds can be formed more easily behind it as the first fold becomes an obstacle, leading to the compression of the crust led by the flow. We want to create a second fold behind the first one (in the direction opposite to the local surface velocity), at a distance equal to the thickness of the folds ℓ.

To do that, we consider the particle at the center of the first fold, which we call P. From its position, we take the position situated at $p' = p + \ell\,\vec{d_\perp}$, where $\vec{d_\perp}$ is the normalized vector orthogonal to the tangent at P and pointing in the direction opposite to velocity. We create a new particle P' at this new position, right behind the first fold. From P', we create a new fold with the same method as before.

When chaining particles from P' with this approach, it can happen that a particle we want to place is too close from the previous fold. In that case, we rather place the new particle so that the currently created fold is locally parallel to the previous fold, as shown in Figure 5.7-right.

5.5.2 Fold dynamics

5.5.2.1 Motion of a fold

Once a fold is created, each particle in it moves at the surface of the flow, advected by the surface velocity. To allow the fold to keep its linear structure, we prevent neighboring particles from moving too far away from each other thanks to spring forces between them.

Finally, since the cylinders associated with the particles have to fully remain within the lava, the fold cannot get too close to the border of the flow. If, after a time step, the position of a particle does not fulfill this requirement, it comes back to its previous position.

5.5.2.2 Collision between two folds

When following folds are created after a first one, they are already pretty close to each other. As the velocity around the first folding is generally lower than around the next ones, the folds tend to collide, as can be observed on the images of real lava flows. Since the motion of the folds is entirely handled by the particles, we model these collisions between the particles of the different folds.

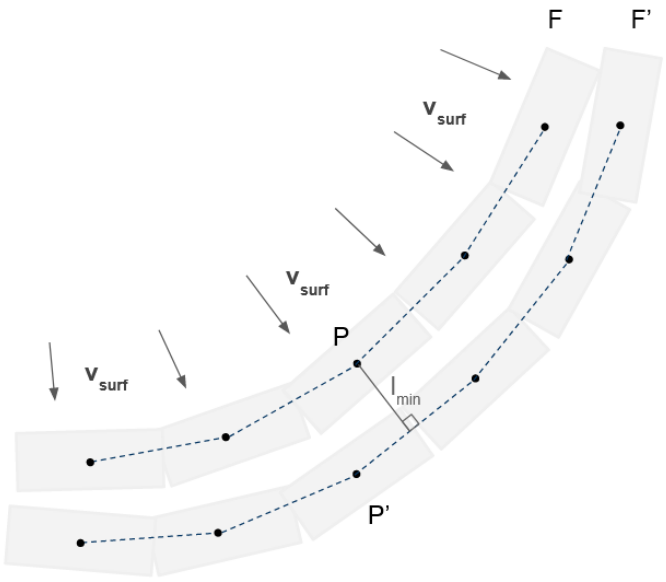

Figure 5.8: Collision detection between two neighbouring folds. Here, F is moving towards F'.

We detect collisions at each time step as follows. We consider two neighbouring folds F and F' and every particle P of F, one after the other. We find the closest particle P' of F' and l_{min} the distance between P and F', as shown in Figure 5.8. If l_{min} is lower than $\frac{\ell_F}{2} + \frac{\ell_{F'}}{2}$, a collision occurs.

Let us consider that F is moving towards F'. As F gets blocked by F', its motion changes to fits the motion of F'. Therefore, we set the velocity of P to the velocity of P', the closest particle in F'. We also

change the position of P if l_{min} is lower than $\frac{l_F}{2} + \frac{l_{F'}}{2}$, to set it to this value, and keep the minimum distance between the two folds.

5.5.2.3 Extended collisions with ascending motion

The method described above is the basic algorithm to make the neighbouring folds collide and then remain in contact, which is plausible in highly viscous materials. However, in the video I linked in the introduction of this chapter, we can see the folds going above or under their neighbours. They can reach a higher position or sink back within the flow under the constraint of a big compression. In this subsection, I describe an alternative way to handle the collisions to better reproduce this behaviour, which is still a work in progress. The results shown in the next section will use the previous algorithm, which is more stable at the moment.

As folds have a cylindrical shape on their visible part, when they collide they may start rolling on each other - as gears would do.

We detect the collision between two particles P and P' of different folds with the same method as before, where P is moving towards P'. We consider the collision between the two cylinders associated to these particles. Using the distance d_{xy} between them, we compute the new height d_z of P such that:

$$d_{xy}^2 + d_z^2 = (\frac{l_F}{2} + \frac{l_{F'}}{2})^2 \tag{5.5}$$

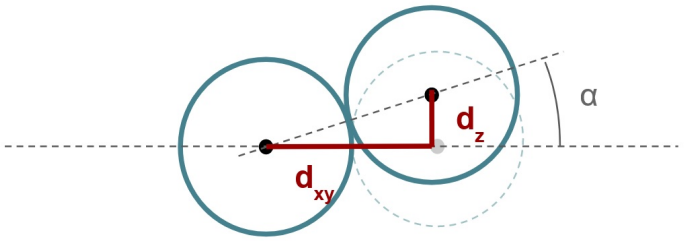

Figure 5.9: Ascending collision between two folds. The fold on the right is moving towards the one on the left. As the horizontal distance between the two particles becomes lower than the sum of their two radii, we compute the new height of the fold on the right to maintain the required distance. This gives us the angle defining its rotation.

This ensures contact between the two cylinders. This equation has two solutions: a positive and a negative one. In the first case, F will go above F', whereas in the second case, F will sink under F'. To decide which solution to choose, we consider several scenarios:

- If P is already above P' before the collision, it will stay above and we choose the positive solution. Similarly, if P is already under P', we choose the negative solution.

- If P and P' are at the same height, we check the neighbours of P are already going above or under F'. If they are already going up or down, P will go in the same direction.

- If none of the neighbours are going above or under F', or if different neighbours are already going in different directions, we randomly choose between the positive and negative solutions.

We also change the orientation of the cylinder around P, with an angle equal to the angle of rotation between the two cylinders α.

Finally, the collision should make the second fold slow down to eventually match the motion of the first fold. To model that, we slowly decrease the velocity of P until it becomes equal to the velocity of P'.

There is still a set of open questions that will need to be tackled in order to achieve the final results:

- The case where a part of a fold F is going above F' and another part underneath F' happens in videos of the real phenomena, and is also possible in our model. We still need to ensure a smooth transition between the upper and lower parts of F, so that it does not collide with F'.

- The case I described in this section took only two neighbouring folds into consideration; but if F rolls over F', it will have an effect on its other neighbour, which may get entrained by F.

5.6 Visualization and results

Test scenarios. For showcasing our current results, we set three test terrains (see Figure 5.10) to enhance how our lava simulation behaves in different situations:

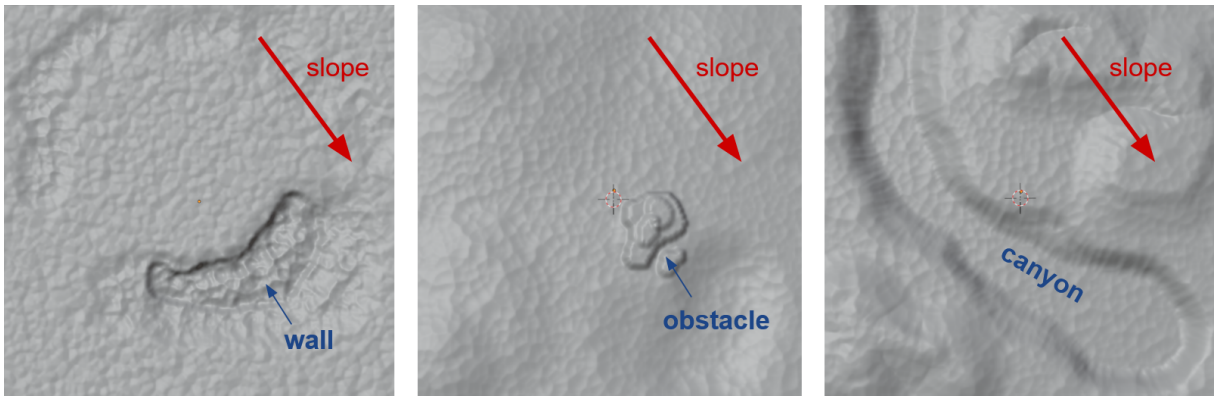

Figure 5.10: Our three test terrains with the direction of the slope going downwards, which is the direction that our lava flows will follow. These are preliminary terrains and will be refined. Left: A large obstacle that will block the lava. Middle: A small obstacle that will make the flow split. Right: A canyon dug in the ground that will guide the flow.

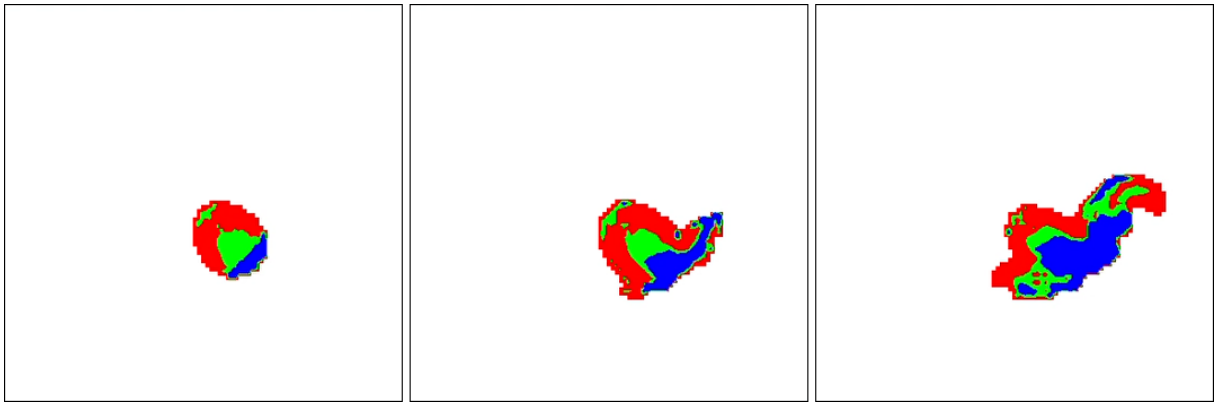

Figure 5.11: Three frames of a lava flow animation where the flow is blocked by a large obstacle. Color code: red represents a crust thinner than 8mm; green is for a crust between 8mm and 2cm; blue is for a crust thicker than 2cm.

1. A slope with a long obstacle at the end to block the lava flow, shown in Figure 5.10-left. Three frames of the simulation of a flow on this terrain can be seen on Figure 5.11. The source is in the middle of the image, in the upper left part of the flow. On the middle frame, the lava gets blocked by the obstacle and begins spreading on the sides. We can observe the crust getting thicker on a larger region with time. On the right of the last frame, the flow finally reaches the end of the obstacle, and liquid lava with a thinner crust begins flowing again from there;

2. A slope with a smaller obstacle that will split the lava flow in two parts, shown on Figure 5.10-middle, and which gives the lava flow animation on Figure 5.12. The lava hits the obstacle on the first frame, and gets split in two separate flows that eventually mix together again on the next frame. Like on the first example, we can observe a bigger region of thicker crust just before the obstacle, and the crust is thinner on the sides and after the split;

3. And a slope with a canyon that will guide the lava to form a river, that we can see on Figure 5.10-right. Figure 5.13 shows the animation of a lava flow in this canyon: the lava river is guided by the terrain. Crust is thicker in the middle of the flow and thinner on the sides, which reproduces what we see on pictures.

3D previsualization of the lava flow. We previsualize our lava flow simulation using a real-time interactive interface, letting us add lava sources where we want on the terrain.

Figure 5.14 shows a visualization of the different lava flow parameters we discussed in Section 5.4, namely:

- The velocity field;

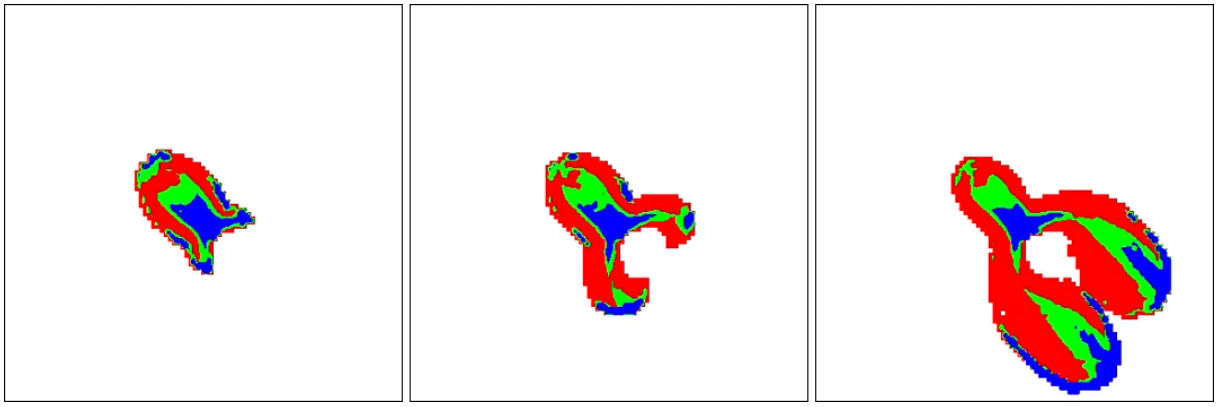

Figure 5.12: Three frames of a lava flow animation where the flow is split by an obstacle. Color code: red represents a crust thinner than 8mm; green is for a crust between 8mm and 2cm; blue is for a crust thicker than 2cm.

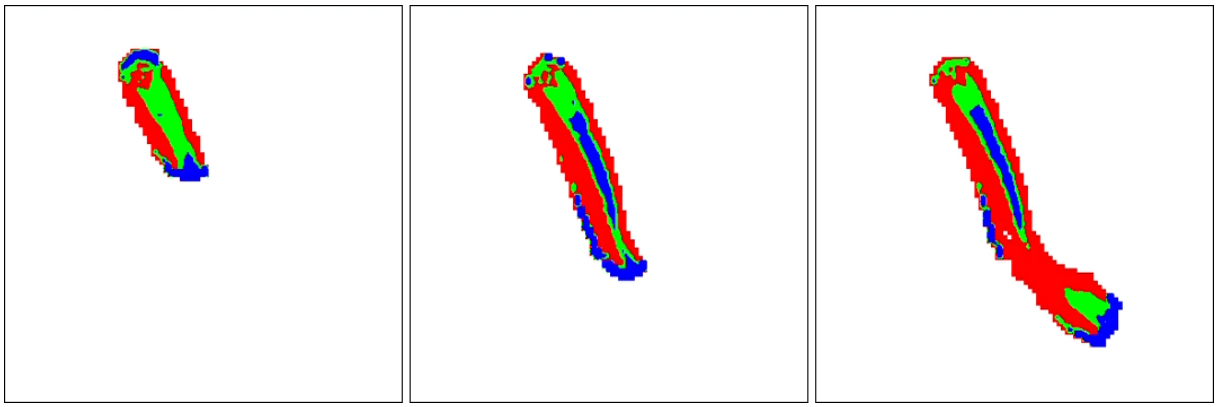

Figure 5.13: Three frames of a lava flow animation where the flow is driven within a long canyon. Color code: red represents a crust thinner than 8mm; green is for a crust between 8mm and 2cm; blue is for a crust thicker than 2cm.

- The lava thickness h_{lava}: this thickness is lower (green) at the border of the flow, and higher (red) near the obstacle where the flow is blocked and the lava accumulates;
- The lava height difference between a cell and its neighbours: this height is higher at the borders, and maximal near the obstacle. Therefore, folds will form in this region;
- The crust thickness h_{crust}, which is higher in two regions: downstream, where a crust has had the time to appear and get thicker; and just behind the lava source, where the lava is static;
- The possibility of a fold to begin: in this test case, the long obstacle causes folds to appear.

Folds visualization. Figure 5.15 shows two different visualizations for the folds at the surface of lava:

- On the left, a real-time previsualization of the folds using chains of cylinders, where each cylinder represents a particle (see Section 5.5). This previsualization is rough but it a good way to see how the surface of the lava looks like, and the system keeps its interactivity.

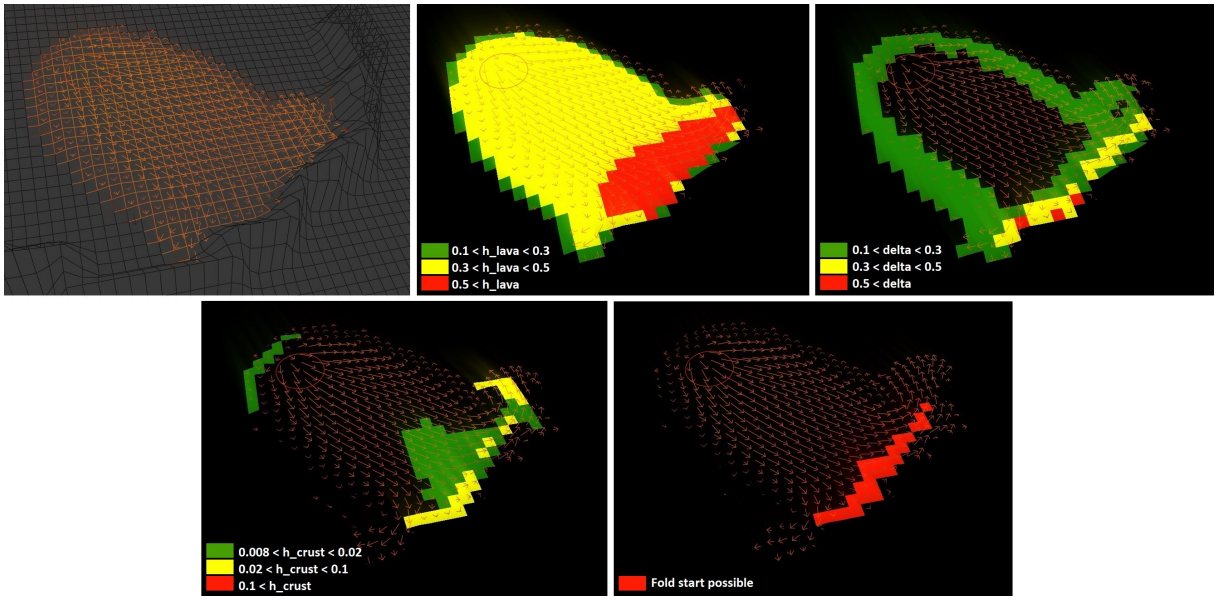

Figure 5.14: Real time visualization of several values on a lava flow for test case 1. The lava source is at the top left of the image, and the lava flows down the slope before hitting a long obstacle. Upper left: velocity field. Upper middle: lava thickness h_{lava}. Upper right: height difference between the lava and the lowest lava height around. Lower left: crust thickness h_{crust}. Lower right: fold start possibility.

- On the right, a work-in-progress visualization of the folds, using implicit surfaces instead of cylinders. This is not a real-time visualization. To get the final results, we still need to work on certain aspects of this representation: we need to make a smooth transition between the folds and the liquid lava surface mesh by deforming it with an implicit deformer; and we need to parametrize this implicit representation so that the transition between the folds is smoother when the crust thickness is thinner.

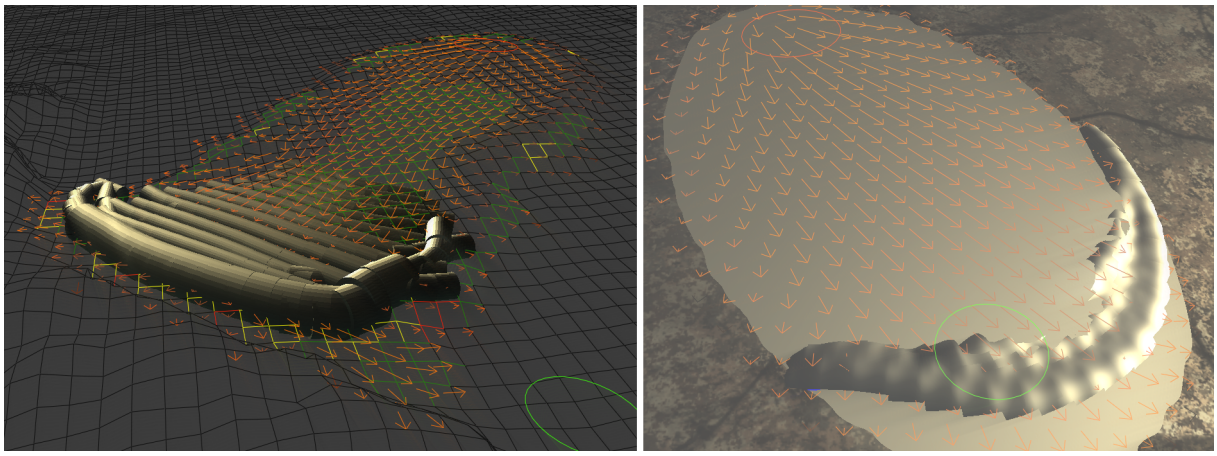

Figure 5.15: Left: Previsualization of folds as cylinders. Right: Implicit surfaces to visualize folds.

Texturing the lava flow. We started a collaboration with Pascal Guehl and Jean-Michel Dischler to extend their approach presented in Guehl et al. (GAD+20) towards texturing lava flows. This problem is very challenging since we would like to synthesize an animated scene, and thus need to handle temporal coherence of the generated textures. The main idea is to define several regions on the flow, attach a texture image to each region, and build semi-procedural textures from this to get smooth texture transitions between the regions. In our case, we define these regions based on the thickness of the crust (see Figure 5.16) using:

Figure 5.16: Separation of the surface of the lava flow in three regions for texturing, depending on the crust thickness.

1. A hot region where the lava at the surface is still liquid and orange;
2. A region where the crust begins to thicken and looks darker, but we can still clearly see orange lava;
3. And a region where the crust gets even thicker, and we can't see much liquid lava anymore.

The main challenge is to get temporal coherence while driving these textures on the surface velocity field, and ensure smooth transitions with the texture of the folds. The latter will be handled separately to maintain their quasi-rigid structure, by attaching texture patches to the particles.

Figure 5.17 shows some first results for the texturing of the lava flow for the test scenarios 2 and 3. We can compare them with Figures 5.13 and 5.12 showing the crust thickness, and notice darker regions where the crust is thicker, as well as yellow/orange regions where the crust is thinner.

5.7 Conclusion and future work

In this chapter, we presented a layered model for the animation of the surface of pahoehoe lava flows. The specificity of these lava flows is the crust that forms at its surface and shows various behaviours

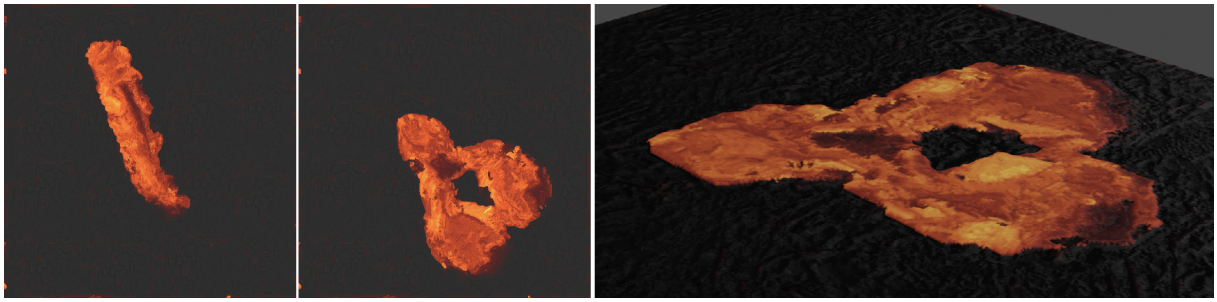

Figure 5.17: Intermediate results for the texturing of the lava flow surface. Left: The flow from Figure 5.13 textured. Middle: The flow from Figure 5.12 textured. Right: The same texture as in the middle image, applied on a mesh.

when solidifying, going from a liquid to a plastic state, before finally becoming rigid. In particular, the crust of the flow can fold under certain conditions.

Our model is built upon a 2D Eulerian simulation of lava, which is our first layer, and from which we get the velocity field of the flow, as well as the lava height. Our second layer computes several surface parameters of the flow, such as the crust thickness and the surface velocity that guides the motion of this crust, and eventually deduces where folds can form, for instance in regions where the lava gets blocked by an obstacle. Our third layer adds a geometric representation for folds at the surface of the flow by chaining cylinders. After the formation of a first fold in a region detected by the second layer, other folds follow behind it. When folds get closer, they collide, and if they get too compressed, they can even roll above or under each other. We previsualize our animation in real-time using a rough representation for folds as cylinders. For our final results, we visualize the folds using implicit surfaces. To texture our lava flows, we define several regions with different general appearances depending on the crust thickness. Our textures are semi-procedural and are generated using different textures attached to each region to get smooth transitions and temporal coherence.

This chapter is still an ongoing work. To get final results, we still need improve several aspects of our model:

- As mentioned at the end of Section 5.5, we need to finish the work on the collisions between folds, to have them properly roll on each other;
- We also need to work on the visualization using an implicit deformer, to parametrize it and obtain smooth transitions between the folds and the surface mesh;
- The work on texturing the lava flow has to be continued to get the final renderings;
- We plan to add a retroaction from our third layer of animation to the first one: we can detect regions where the lava accumulates, leading to high pressures; the latter could break through a

thin crust to start a new flow. In this case, we could automatically add a new lava source to seed a new flow, similar to the ones observed on reference photos (see Figure 5.2-left and right).

Even without the final results, we already identified several limitations of our approach. First, we specifically tackled the surface animation of lava flows, and our method could hardly be used for different materials. Second, our animation focuses on the short-term hardening of the surface of lava flows, and does not handle the long-term behaviour of hardening lava that eventually changes the terrain.

On the long term, we see possible future work in different directions. A first extension would be to add even more diversity in the folds by adding branches to them, or by getting them rigidly attached after some time, once the gaps between them becomes rigid as well. More generally, we could get more diversity in the different structures evolving at the surface of pahoehoe lava flows: indeed, before forming folds, a thin crust can rip if the stretch applied on it is too high, and therefore we can observe small pieces of crust floating of the flow. We could generalized the fold structure to handle this type of objects at the surface of flows.

Another way of improving our animation of pahoehoe lava flows would be to improve the visual aspect and animation of the borders of flow, and especially the front. Indeed, our approach captures neither the round shape that we can see on Figure 5.2-right, nor the swelling of flow under a thin crust, nor the texture generation occurring there as the flow grows.

In our model, we did not tackle the hardening of the flow on the long-term, which can take days or even weeks, and can produces empty chambers under the crust as the liquid lava can continue flowing underneath for a long time. An interesting future work would be to tackle the life of a volcano producing lava flows on the long term, and see how its eruptions change the terrain.

In the future, we would also like to extend our model to tackle user interaction by creating tools to enable the authoring of a lava flow. For instance, the user could decide where they want to see folds form; or they could use an eyedropper tool to sample textures from lava flow videos, and apply them in their animation to get the desired aspect they want.

Our approach could be used to animate other types of flows such as a'a lava flows. Indeed, a lava flow can start as pahoehoe to become a a'a flow later: we could animate even more complex flows, switching between different types and textures of lava flows. Lastly, we could combine this system with the models for the animation of volcanic plumes presented in the previous chapter to get a complete simulator for diverse volcanic eruptions.

Chapter 6

Conclusion

Throughout this thesis, I introduced new methods for plausible, fast, and controllable animation of volcanic eruptions. In this conclusion, I will present our different contributions before presenting some general research perspectives.

6.1 Contributions

To animate different volcanic phenomena, we used a mixed set of hybrid approaches depending on the type of volcanic eruption and which specific effects are still missing or hard to model in the current state-of-the-art in Graphics. The three different projects developed in this thesis satisfy the requirements and constraints proposed at the beginning: the plausibility of the results to obtain animations close to reality in terms of geometric resemblance and temporal dynamics; the rapidity of the computation, to enable our systems to be interactive; and the controllability of our models with light and easy to understand approaches and user control when possible.

All the work presented in this thesis mixes a simple simulation with procedural models to add details. It is a good way to get fast interactive systems, and we can easily control the procedural part to address specific natural phenomena, and get plausible results. However, it is limited in terms of phenomena we can represent with each system: every time we want to add a new sub-phenomena, we have to extend our procedural model; and the minimalistic simulation may give insufficient precision for the animation of certain sub-phenomena. With these layered models, I also used a phenomenological approach by reproducing complex phenomena described in the volcanology literature and visible on photographs and videos, but that have not been much tackled in Graphics.

Volcanic plumes and pyroclastic flows. We developed a new multi-layer model to consistently animate plausible volcanic plumes and pyroclastic flows in real time. For our model, we combine a fast minimalist Lagrangian simulation to animate the plume that we discretize into slices, and a procedural model to add fractal details with a multi-scale set of spheres to reproduce the billowing aspect of the turbulent plume. Our simulation takes phenomena such as air entrainment and large density variations into account. These effects are crucial to depict the specific behaviour of volcanic plumes, and enables our system to detect the formation of pyroclastic flows, which happens when the volcanic column is too heavy relative to its ejection speed and cannot rise. Our dynamic plume model can be extended to also represent pyroclastic flows. In this case, the volume of the flow moving downward along the terrain is simulated using discrete Lagrangian particles that loose density along time. Groups of lightweight particles in this flow can eventually lead to the emergence of secondary plumes that can then be simulated in reusing our approach. We validate our results by comparing the evolution of speed and density of our volcanic plumes to reference curves from the volcanology literature. We give control to the user by letting them change the parameters of the eruption (initial speed and density of the plume, wind speed) on the fly during the simulation.

Clouds and volcanic plumes in the atmosphere. To animate complex skyscapes and simulate volcanic plumes interacting with a surrounding atmosphere, we presented a fast multi-layer weather model. We represent the atmosphere as a set of horizontal layers and simulate different meteorological phenomena in them, as well as an interaction with the terrain, to dynamically obtain plausible cloud formation and animation. We extended this weather model as well as our plume model, by coupling the two systems, with the objective of animating volcanic columns in a complex environment, representing new clouds that appear around the plume and animating extra phenomena such as ash rain.

Lava flows. Finally, we developed a layered model, that works interactively, to consistently animate the surface of pahoehoe lava flows that can be observed in Hawaii. The surface of such fluid lava flows hardens at the contact with air, and can form various complex structures such as folds. We used a 2D Eulerian simulation computing an average motion for the underlying lava flow used as a base layer and we build our surface model on it. In a first layer, we started by computing different surface parameters, such as the surface velocity and the crust thickness, to know where a crust appears on the flow, and eventually eventually create folds. In our second layer, we added a geometric representation for folds at the surface of lava as chains of cylinders driven by the surface velocity of the lava flow. Thanks to their geometrical representation, we can handle efficiently collisions between neighbouring rigid fold crusts, enabling them to eventually roll on each other. The lava crust is to be visualized via a surface

deformation based on an implicit deformer, and temporally coherent semi-procedural textures.

6.2 Perspectives

More research can be done on the animation and authoring of natural phenomena such as volcanic eruptions. In this last section, I present some promising directions and challenges for research.

Towards more diversity in volcano animation. In our work, we tackled very different types of volcanic eruptions and ejections, from volcanic plumes and pyroclastic flows to very fluid lava flows. In the future, I believe that even more details and diversity can be added to the animation of these phenomena. For instance, there is room to improve the details on volcanic columns, the diversity of shapes of volcanic plumes. For the animation of lava flows, there are many phenomena on pahoehoe lava that have never been tackled in CG, such as the ripping of the crust and the formation of new lobes of lava flow. Moreover, other types of volcanic phenomena could be animated with novel specific models, such as submarine volcanoes forming pillow lavas, or the formation of hexagonal basalt columns.

Simulating how volcanoes affect the world. In this thesis, I focused on the short-term animation of volcanic eruptions, as the most visual spectacular phenomena happen during them. However, volcanic eruptions may have long-term effects on the weather and the terrain that would be interesting to simulate in CG, to model realistic terrains and skyscapes. Indeed, big volcanic plumes can cause a global cooling of the atmosphere and have an effect on the color of the sky, which would be interesting to integrate in weather systems. They also affect the geometry of the terrain because of ash deposits that can increase the height of the terrain by several meters; and pyroclastic flows may also change its aspect by destroying the vegetation on their ways. Eruptions can also cause landslides around the vent, and form calderas if the magma chamber under the volcano collapses. Lava flows also change the geometry of the terrain after each eruption, and submarine volcanoes can even form islands on the long term. All of these effects on terrain would be an interesting target for future research on terrain generation.

Authoring of volcanic eruptions. Finally, while I focused on the visual simulation of volcanic phenomena giving indirect control to users, I believe that creating tools for direct control is crucial in the future. Indeed, the existing tools are limited and do not give easy ways to animate plausible volcanic plumes and lava flows. The results produced with these tools are either unrealistic, or take the artists a lot of time and efforts. Possible directions for research in this field are, for instance: sketch-based modeling and animation, enabling users to draw the trajectory of volcanic plumes, pyroclastic flows and lava flows, and the shape of plumes; eyedropper tools to sample textures from videos and apply them

on lava flows; and other selection tools to choose where to form secondary columns from pyroclastic flows, or where to form folds on pahoehoe lava flows. To help artists authoring volcanic eruptions for games and movies, creating a general volcanic eruption authoring system handling different types of volcanoes would be an objective for the future.

Bibliography

[AIA+12] Nadir Akinci, Markus Ihmsen, Gizem Akinci, Barbara Solenthaler, and Matthias Teschner. Versatile rigid-fluid coupling for incompressible sph. *ACM Trans. Graph.*, 31(4), jul 2012.

[And18] Ryoichi Ando. Shiokaze Framework, 2018. http://research.nii.ac.jp/~rand/shiokaze-en/.

[ANSN06] Alexis Angelidis, Fabrice Neyret, Karan Singh, and Derek Nowrouzezahrai. A controllable, fast and stable basis for vortex based smoke simulation. *SCA*, page 25–32, 2006.

[APKG07] Bart Adams, Mark Pauly, Richard Keiser, and Leonidas J. Guibas. Adaptively sampled particle fluids. *ACM Transactions on Graphics*, 2007.

[AV07] D Arthur and S Vassilvitskii. k-means++: the advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms)*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[Bar92] Rogers Barry. *Mountain weather and climate*. Routledge, 2 edition, 1992.

[BC09] Rogers Barry and Richard Chorley. *Atmosphere, Weather and Climate*. Routledge, 9 edition, 2009.

[BDM+16] Mathias Brousset, Emmanuelle Darles, Daniel Meneveaux, Pierre Poulin, and Benoît Crespin. Simulation and control of breaking waves using an external force model. *Computers & Graphics*, 57:102–111, 2016.

[BN04] Antoine Bouthors and Fabrice Neyret. Modeling Clouds Shape. In Eric Galin and Marc Alexa, editors, *Eurographics (short papers)*, pages –, Grenoble, France, August 2004. Eurographics Association.

[BNL06] Antoine Bouthors, Fabrice Neyret, and Sylvain Lefebvre. Real-time realistic illumination and shading of stratiform clouds. In Eric Galin and Norishige Chiba, editors, *Eurographics Workshop on Natural Phenomena*, Vienne, Austria, September 2006. Eurographics.

[BSS14] Noé Bernabeu, Pierre Saramito, and Claude Smutek. Numerical modeling of non-Newtonian viscoplastic flows: Part II. Viscoplastic fluids and general tridimensional topographies. *International Journal of Numerical Analysis and Modeling*, 11(1):213–228, January 2014. Dedicated to Professor Francisco J. Lisbona on the occasion of his 65th Birthday.

[CAIK+14] C. Cimarelli, M.A. Alatorre-Ibargüengoitia, U. Kueppers, B. Scheu, and D.B. Dingwell. Experimental generation of volcanic lightning. *Geology*, 42(1):79–82, 01 2014.

[Can93] Marie-Paule Cani. An implicit formulation for precise contact modeling between flexible solids. In *20th annual conference on Computer graphics and interactive techniques (SIGGRAPH '93))*, pages 313–320, Anaheim, United States, August 1993. ACM, ACM SIGGRAPH. Published under the name Marie-Paule Gascuel.

[CB15] Steven Carey and Marcus Bursik. *Encyclopedia of Volcanoes - Volcanic Plumes*, chapter 32, pages 571–585. University of Chicago Press, 2015.

[CBC+16] Guillaume Cordonnier, Jean Braun, Marie-Paule Cani, Bedrich Benes, Éric Galin, Adrien Peytavie, and Éric Guérin. Large scale terrain generation from tectonic uplift and fluvial erosion. *Computer Graphics Forum*, 35(2):165–175, 2016.

[CBL+09] Yuanzhang Chang, Kai Bao, Youquan Liu, Jian Zhu, and Enhua Wu. A particle-based method for viscoelastic fluids animation. In *Proceedings of the 16th ACM Symposium on Virtual Reality Software and Technology - VRST '09*, page 111, Kyoto, Japan, 2009. ACM Press.

[CBP05] Simon Clavet, Philippe Beaudoin, and Pierre Poulin. Particle-based viscoelastic fluid simulation. *Symposium on Computer Animation*, 2005.

[CCB+18] Guillaume Cordonnier, Marie-Paule Cani, Bedrich Benes, Jean Braun, and Eric Galin. Sculpting Mountains: Interactive Terrain Modeling Based on Subsurface Geology. *IEEE Transactions on Visualization and Computer Graphics*, 24(5):1756–1769, May 2018.

[CGG+17] Guillaume Cordonnier, Eric Galin, James Gain, Bedrich Benes, Eric Guérin, Adrien Peytavie, and Marie-Paule Cani. Authoring landscapes by combining ecosystem and terrain erosion simulation. *ACM Trans. Graph.*, 36(4):134:1–134:12, July 2017.

[CK00] G.-H. Cottet and P. D. Koumoutsakos. *Vortex Methods: Theory and Practice*. Cambridge University Press, 2000.

[CMCM11] Yu Chung-Min and Wang Chung-Ming. An effective framework for cloud modeling, rendering, and morphing. *Journal of Information Science and Engineering*, 27(3):891–913, May 2011.

[CMVHT02] Mark Carlson, Peter J. Mucha, R. Brooks Van Horn, and Greg Turk. Melting and flowing. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '02, page 167–174, New York, NY, USA, 2002. Association for Computing Machinery.

[CS15] Eric Chahi and François Sahy. Volcano simulator, 2015.

[CWWY21] Qian Chen, Yue Wang, Hui Wang, and Xubo Yang. Data-driven simulation in fluids animation: A survey. *Virtual Reality & Intelligent Hardware*, 3(2):87–104, 2021. Special issue on simulation and interaction of fluid and solid dynamics.

[DC96] Mathieu Desbrun and Marie-Paule Cani. Smoothed particles: A new paradigm for animating highly deformable bodies. *Computer Animation and Simulation '96 (Proceedings of EG Workshop on Animation and Simulation)*, 1996.

[DC99] Mathieu Desbrun and Marie-Paule Cani. Space-time adaptive simulation of highly deformable substances. Technical report, INRIA, 1999.

[DG95] Mathieu Desbrun and Marie-Paule Gascuel. Animating soft substances with implicit surfaces. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, page 287–290, New York, NY, USA, 1995. Association for Computing Machinery.

[DG17] Rui P.M. Duarte and Abel J.P. Gomes. Real-time simulation of cumulus clouds through skewt/logp diagrams. *Computers & Graphics*, 67:103 – 114, 2017.

[DKY+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*, SIGGRAPH '00, pages 19–28, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[DR06] F. Dobran and J. I. Ramos. Global volcanic simulation: Physical modeling, numerics, and computer implementation. *Developments in Volcanology*, 8:311–372, 2006.

[Dre01] PDI Dreamworks. Chapter 'Flames and Dragon Fireballs' in 'Shrek: The Story Behind the Screen'. *ACM SIGGRAPH Course Notes*, 2001.

[DSY10] Yoshinori Dobashi, Yusuke Shinzo, and Tsuyoshi Yamamoto. Modeling of clouds from a single photograph. *Computer Graphics Forum*, 29(7):2083–2090, 2010.

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

[EFFM02] Douglas Enright, Ronald Fedkiw, Joel Ferziger, and Ian Mitchell. A Hybrid Particle Level Set Method for Improved Interface Capturing. *Journal of Computational Physics*, 183(1):83–116, November 2002.

[EMF02] Douglas Enright, Stephen Marschner, and Ronald Fedkiw. Animation and rendering of complex water surfaces. *ACM Trans. Graph.*, 21(3):736–744, jul 2002.

[EMP+02] David S. Ebert, F. Kenton Musgrave, Darwyn Peachey, Ken Perlin, and Steven Worley. *Texturing and Modeling: A Procedural Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2002.

[FBDY15] Charles Welton Ferreira Barbosa, Yoshinori Dobashi, and Tsuyoshi Yamamoto. Adaptive cloud simulation using position based fluids. *Computer Animation and Virtual Worlds*, 26(3-4):367–375, 2015.

[FF01] Nick Foster and Ronald Fedkiw. Practical animation of liquids. SIGGRAPH '01, page 23–30, New York, NY, USA, 2001. Association for Computing Machinery.

[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*, SIGGRAPH '97, page 181–188, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[FR86] Alain Fournier and William T. Reeves. A simple model of ocean waves. *SIGGRAPH Comput. Graph.*, 20(4):75–84, aug 1986.

[FSJ01] Ronald Fedkiw, Jos Stam, and Henrik Wann Jensen. Visual simulation of smoke. *SIGGRAPH*, pages 15–22, 2001.

[GAD+20] P. Guehl, R. Allègre, J.-M. Dischler, B. Benes, and E. Galin. Semi-procedural textures using point process texture basis functions. *Computer Graphics Forum*, 39(4):159–171, 2020.

[Gar85] Geoffrey Y. Gardner. Visual simulation of clouds. In *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '85, pages 297–304, New York, NY, USA, 1985. ACM.

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

[GDAB+17] Ignacio Garcia-Dorado, Daniel G. Aliaga, Saiprasanth Bhalachandran, Paul Schmid, and Dev Niyogi. Fast weather simulation for inverse procedural design of 3d urban models. *ACM Trans. Graph.*, 36(4), April 2017.

[GDG+17] Eric Guérin, Julie Digne, Eric Galin, Adrien Peytavie, Christian Wolf, Bedrich Benes, and Benoît Martinez. Interactive example-based terrain authoring with conditional generative adversarial networks. *ACM Trans. Graph.*, 36(6), nov 2017.

[GDGP16] Eric Guérin, Julie Digne, Eric Galin, and Adrien Peytavie. Sparse representation of terrains for procedural modeling. *Computer Graphics Forum (proceedings of Eurographics 2016)*, 35(2):177–187, 2016.

[GGV+19] Jonathan Gagnon, Julián E. Guzmán, Valentin Vervondel, François Dagenais, David Mould, and Eric Paquette. Distribution Update of Deformable Patches for Texture Synthesis on the Free Surface of Fluids. *Computer Graphics Forum*, 38(7):491–500, October 2019.

[GHB+20] Christoph Gissler, Andreas Henne, Stefan Band, Andreas Peer, and Matthias Teschner. An implicit compressible sph solver for snow simulation. *ACM Trans. Graph.*, 39(4), jul 2020.

[GL94] J. Gilbert and Stephen Lane. The origin of accretionary lapilli. *Bulletin of Volcanology*, 56:398–411, 11 1994.

[GLCC17] J. Gain, H. Long, G. Cordonnier, and M.-P. Cani. Ecobrush: Interactive control of visually consistent large-scale ecosystems. *Computer Graphics Forum*, 36(2):63–73, 2017.

[GLS+20] Geoffrey Guingo, Frédéric Larue, Basile Sauvage, Nicolas Lutz, Jean-Michel Dischler, and Marie-Paule Cani. Content-aware texture deformation with dynamic control. *Computers and Graphics*, 91:95–107, October 2020.

[GM77] R. A. Gingold and J. J. Monaghan. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society*, 1977.

[GN17] Prashant Goswami and Fabrice Neyret. Real-time landscape-size convective clouds simulation and rendering. In *Proceedings of Workshop on Virtual Reality Interaction and Physical Simulation*, 2017.

[Har63] F.H. Harlow. The particle-in-cell method for numerical solution of problems in fluid dynamics. 1963.

[HHP+21] Jorge Alejandro Amador Herrera, Torsten Hädrich, Wojtek Pałubicki, Daniel T. Banuti, Sören Pirk, and Dominik L. Michels. Weatherscapes: Nowcasting heat transfer and water continuity. *ACM Transaction on Graphics*, 40(6), 12 2021.

[HMP+20] Torsten Hadrich, Milosz Makowski, Wojtek Palubicki, Daniel T. Banuti, Soren Pirk, and Dominik L. Michels. Stormscapes: Simulating Cloud Dynamics in the Now. *ACM TOG, Proc. SIGGRAPH*, 2020.

[HNC02] Damien Hinsinger, Fabrice Neyret, and Marie-Paule Cani. Interactive Animation of Ocean Waves. In Stephen N. Spencer, editor, *ACM-SIGGRAPH/EG Symposium on Computer Animation (SCA'02)*, pages 161 – 166, San Antonio, United States, July 2002. ACM SIGGRAPH.

[HQT+21] Libo Huang, Ziyin Qu, Xun Tan, Xinxin Zhang, Dominik L. Michels, and Chenfanfu Jiang. Ships, splashes, and waves on a vast ocean. *ACM Trans. Graph.*, 40(6), dec 2021.

[HW65] Francis H. Harlow and J. Eddie Welch. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *The Physics of Fluids*, 8(12):2182–2189, 1965.

[Jau91] Claude Jaupart. Effects of compressibility on the flow of lava. *Bulletin of Volcanology*, 54:1–9, 1991.

[Jau96] Claude Jaupart. Physical models of volcanic eruptions. *Chemical Geology*, 128:217–227, 1996.

[JWL+08] Manish Rohan James, L. Wilson, Stephen J. Lane, Jennie S Gilbert, Tamsin A. Mather, R. G. Harrison, and R. S. Martin. Electrical charging of volcanic plumes. *Space Science Reviews*, 137:399–418, 2008.

[KAG+06] Richard Keiser, Bart Adams, Leonidas J. Guibas, Philip Dutré, and Mark Pauly. Multiresolution particle-based fluids. Technical report, ETH, 2006.

[KAK+06] Vivek Kwatra, David Adalsteinsson, Nipun Kwatra, Mark Carlson, and Ming C. Lin. Texturing fluids. In *ACM SIGGRAPH 2006 Sketches*, SIGGRAPH '06, page 63–es, New York, NY, USA, 2006. Association for Computing Machinery.

[Kap03] Alan Kapler. Avalanche! Snowy FX for XXX. In *ACM SIGGRAPH 2003 Sketches & Applications*. ACM, 2003.

[KHW+22] Byungsoo Kim, Xingchang Huang, Laura Wuelfroth, Jingwei Tang, Guillaume Cordonnier, Markus Gross, and Barbara Solenthaler. Deep Reconstruction of 3D Smoke Densities from Artist Sketches. *Computer Graphics Forum*, 41(2):97–110, May 2022.

[KJ97] Édouard Kaminski and Claude Jaupart. Expansion and quenching of vesicular magma fragments in plinian eruptions. *Journal of Geophysical Research: Solid Earth*, 102(B6):12187–12203, 1997.

[KMM+17] Simon Kallweit, Thomas Müller, Brian Mcwilliams, Markus Gross, and Jan Novák. Deep scattering: Rendering atmospheric clouds with radiance-predicting neural networks. *ACM Trans. Graph.*, 36(6):231:1–231:11, November 2017.

[KVH84] James T. Kajiya and Brian P Von Herzen. Ray tracing volume densities. *SIGGRAPH Computer Graphics*, 1984.

[Lin68] Aristid Lindenmayer. Mathematical models for cellular interactions in development i. filaments with one-sided inputs. *Journal of Theoretical Biology*, 18(3):280–299, 1968.

[LJW06] Shengjun Liu, Xiaogang Jin, and Charlie C. L. Wang. Target shape controlled cloud animation. In Tomoyuki Nishita, Qunsheng Peng, and Hans-Peter Seidel, editors, *Advances in Computer Graphics*, pages 578–585, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[LJWcH07] Shengjun Liu, Xiaogang Jin, Charlie C.L. Wang, and Kin chuen Hui. Ellipsoidal-blob approximation of 3d models and its applications. *Computers & Graphics*, 31(2):243 – 251, 2007.

[LLC+10] A. Lagae, S. Lefebvre, R. Cook, T. DeRose, G. Drettakis, D.S. Ebert, J.P. Lewis, K. Perlin, and M. Zwicker. A survey of procedural noise functions. *Computer Graphics Forum*, 29(8):2579–2600, 2010.

[LRC+22] Maud Lastic, Damien Rohmer, Guillaume Cordonnier, Claude Jaupart, Fabrice Neyret, and Marie-Paule Cani. Interactive simulation of plume and pyroclastic volcanic ejections. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 5(1):1–15, May 2022.

[Luc77] L. B. Lucy. A numerical approach to the testing of the fission hypothesis. *The Astronomical Journal*, 1977.

[MDH07] Xing Mei, Philippe Decaudin, and Bao-Gang Hu. Fast Hydraulic Erosion Simulation and Visualization on GPU. In Marc Alexa, Steven J. Gortler, and Tao Ju, editors, *PG '07 - 15th Pacific Conference on Computer Graphics and Applications*, Pacific Graphics 2007, pages 47–56, Maui, United States, October 2007. IEEE.

[MDN02] Ryoichi Mizuno, Yoshinori Dobashi, and Tomoyuki Nishita. Volcanic smoke animation using CML. *Proc. Int. Computer Symposium*, 2002.

[MFi21] MFiX. Multiphase flow with interface exchange, 2021.

[MIY02] R. MIYAZAKI. Simulation of cumuliform clouds based on computational fluid dynamics. *EUROGRAPHICS 2002 Short Presentations*, pages 405–410, 2002.

[MTT56] B. R. Morton, G. I. Taylor, and J. S. Turner. Turbulent gravitational convection from maintained and instantaneous sources. *Proc. R. Soc. London*, 234:1—–23, 1956.

[MYND01] Ryo Miyazaki, Satoru Yoshida, Tomoyuki Nishita, and Yoshinori Dobashi. A method for modeling clouds based on atmospheric fluid dynamics. In *Proceedings of the 9th Pacific Conference on Computer Graphics and Applications*, PG '01, pages 363–, Washington, DC, USA, 2001. IEEE Computer Society.

[Ney97] Fabrice Neyret. Qualitative Simulation of Cloud Formation and Evolution. In *Eurographics Workshop on Computer Animation and Simulation*, Budapest, Hungary, 1997.

[Ney03] Fabrice Neyret. Advected Textures. In D. Breen and M. Lin, editors, *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pages 147–153, San diego, United States, July 2003. Eurographics Association.

[NKL+07] Rahul Narain, Vivek Kwatra, Huai-Ping Lee, Theodore Kim, Mark Carlson, and Ming C. Lin. Feature-guided dynamic texture synthesis on continuous flows. *Eurographics Symposium on Rendering*, 2007.

[OK12] Jens Orthmann and Andreas Kolb. Temporal blending for adaptive sph. *Computer Graphics Forum*, 2012.

[PC01] Frank Perbet and Marie-Paule Cani. Animating Prairies in Real-Time. In John F. Hughes and Carlo H. Séquin, editors, *ACM-SIGGRAPH Symposium on Interactive 3D Graphics (I3D'01)*, pages 103–110, Chapel Hill, United States, March 2001. ACM Press.

[PCCP05] Sanjit Patel, Anson Chu, Jonathan Cohen, and Frederic Pighin. Fluid simulation via disjoint translating grids. *ACM SIGGRAPH Sketches*, pages 139—-es, 2005.

[Per85] Ken Perlin. An image synthesizer. *SIGGRAPH Comput. Graph.*, 19(3):287–296, jul 1985.

[PGP+19] Axel Paris, Eric Galin, Adrien Peytavie, Eric Guérin, and James Gain. Terrain Amplification with Implicit 3D Features. *ACM Transactions on Graphics*, 38(5), 2019.

[PPG+20] Axel Paris, Adrien Peytavie, Eric Guérin, Jean-Michel Dischler, and Eric Galin. Modeling Rocky Scenery using Implicit Blocks. *The Visual Computer*, 36(10):2251–2261, 2020.

[PSA+98] Roger A. Pielke, . Sr, 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. *Global Change Biology*, 4(5):461–475, 1998.

[PT80] Donald W. Peterson and Robert I. Tilling. Transition of basaltic lava from pahoehoe to aa, kilauea volcano, hawaii: Field observations and key factors. *Journal of Volcanology and Geothermal Research*, 7(3):271–293, 1980.

[RBD+18] David A. Randall, Cecilia M. Bitz, Gokhan Danabasoglu, A. Scott Denning, Peter R. Gent, Andrew Gettelman, Stephen M. Griffies, Peter Lynch, Hugh Morrison, Robert Pincus, and John Thuburn. 100 years of earth system model development. *Meteorological Monographs*, 59:12.1–12.66, 2018.

[Rey87] Craig W. Reynolds. Flocks, herds and schools: A distributed behavioral model. *SIGGRAPH Comput. Graph.*, 21(4):25–34, aug 1987.

[Rib04] N. M. Ribe. Coiling of viscous jets. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 460(2051):3223–3239, 2004.

[RPC+10] Damien Rohmer, Tiberiu Popa, Marie-Paule Cani, Stefanie Hahmann, and Sheffer Alla. Animation Wrinkling: Augmenting Coarse Cloth Simulations with Realistic-Looking Wrinkles. *ACM Transactions on Graphics*, 29(5):157, December 2010. Special Issue: SIGGRAPH Asia 2010.

[SAC+99] Dan Stora, Pierre-Olivier Agliati, Marie-Paule Cani, Fabrice Neyret, and Jean-Dominique Gascuel. Animating lava flows. *Graphics Interface (GI'99) Proceedings*, 1999.

[SBRS10] Marc Stiver, Andrew Baker, Adam Runions, and Faramarz Samavati. Sketch based volumetric clouds. In Robyn Taylor, Pierre Boulanger, Antonio Krüger, and Patrick Olivier, editors, *Smart Graphics*, pages 1–12, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[SCC+16] Y. J. Suzuki, A. Costa, M. Cerminara, T. Esposti Ongaro, M. Herzog, A. R. Van Eaton, and L. C. Denby. Inter-comparison of three-dimensional models of volcanic plumes. *Journal of Volcanology and Geothermal Research*, 326:26–42, 2016.

[SCLP04] Maurya Shah, Jonathan M. Cohen, Sanjit Patel Penne Lee, and Frédéric Pighin. Extended galilean invariance for adaptive fluid simulation. *Symp. on Computer Animation*, pages 213—-221, 2004.

[SF95] Jos Stam and Eugene Fiume. Depicting fire and other gaseous phenomena using diffusion processes. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, page 129–136, New York, NY, USA, 1995. Association for Computing Machinery.

[SG11] Barbara Solenthaler and Markus Gross. Two-scale particle simulation. *ACM Transactions on Graphics*, 2011.

[SGVE+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. *Geophysical Research Letters*, 48(11):e2020GL092323, 2021. e2020GL092323 2020GL092323.

[Sid21] SideFX. Houdini, 2021.

[SK10] Y. J. Suzuki and T. Koyaguchi. Numerical determination of the efficiency of entrainment in volcanic eruption columns. *Geophysical Research Letters*, 37(5):L05302–, 2010.

[SK15] Y. J. Suzuki and T. Koyaguchi. Effects of wind on entrainment efficiency of wind on entrainment efficiency in volcanic plumes. *Journal of Geophysical Research: Solid Earth*, 120:6122–6140, 2015.

[Spa86] R. Sparks. The dimensions and dynamics of volcanic eruption columns. *Bulletin of Volcanology*, 48:3–15, 1986.

[SSC+13] Alexey Stomakhin, Craig Schroeder, Lawrence Chai, Joseph Teran, and Andrew Selle. A material point method for snow simulation. *ACM Trans. Graph.*, 32(4), jul 2013.

[SSEH03] 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*, SCA '03, pages 160–166, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[SSJ+14] Alexey Stomakhin, Craig Schroeder, Chenfanfu Jiang, Lawrence Chai, Joseph Teran, and Andrew Selle. Augmented MPM for phase-change and varied materials. *ACM Transactions on Graphics*, 33(4):1–11, July 2014.

[Sta99] Jos Stam. Stable fluids. *SIGGRAPH*, pages 121–128, 1999.

[TGL+05] C. Textor, H.-F. Graf, A. Longo, A. Neri, T. Esposti Ongaro, P. Papale, C. Timmreck, and G. G.J. Ernst. Numerical simulation of explosive volcanic eruptions from the conduit flow to global atmospheric scales. *Annals of Geophysics*, 48, 2005.

[TKP13] Nils Thuerey, Theodore Kim, and Tobias Pfaff. Turbulent Fluids. *ACM SIGGRAPH Course Notes*, 2013.

[TL19] Tetsuya Takahashi and Ming C. Lin. A geometrically consistent viscous fluid solver with two-way fluid-solid coupling. *Computer Graphics Forum*, 38(2):49–58, 2019.

[Ton91] D. Tonnesen. Modeling liquids and solids using thermal particles. In *Proceedings of Graphics Interface '91*, GI '91, pages 255–262, Toronto, Ontario, Canada, 1991. Canadian Man-Computer Communications Society.

[TPF89] D. Terzopoulos, J. Platt, and K. Fleischer. Heating and melting deformable models (from goop to glop). In *Proceedings of Graphics Interface '89*, GI '89, pages 219–226, Toronto, Ontario, Canada, 1989. Canadian Man-Computer Communications Society.

[VEMH+15] Alexa Van Eaton, Larry Mastin, Michael Herzog, Hans Schwaiger, David Schneider, Kristi Wallace, and Amanda Clarke. Hail formation triggers rapid ash aggregation in volcanic plumes. *Nature Communications*, 6, 08 2015.

[VGL+20] Ulysse Vimont, James Gain, Maud Lastic, Guillaume Cordonner, Babatunde Abiodun, and Marie-Paule Cani. Interactive meso-scale simulation of skyscapes. *CGF, Proc. Eurographics*, 39(2), 2020.

[WBC08] Jamie Wither, Antoine Bouthors, and Marie-Paule Cani. Rapid sketch modeling of clouds. In Christine Alvarado and Marie-Paule Cani, editors, *Eurographics Workshop on Sketch-Based Interfaces and Modeling (SBIM)*, pages 113–118, Annecy, France, June 2008. Eurographics Association.

[WBDY15] Charles Welton, Ferreira Barbosa, Yoshinori Dobashi, and Tsuyoshi Yamamoto. Adaptive cloud simulation using position based fluids. *Computer Animation and Virtual Worlds*, 26(3-4):367–375, 2015.

[WCGG18] Antoine Webanck, Yann Cortial, Eric Guérin, and Eric Galin. Procedural Cloudscapes. *Computer Graphics Forum*, 37(2), 2018.

[WGM14] Chen Wei, James Gain, and Patrick Marais. Interactive 3d cloud modelling with a brush painting interface. In *Proceedings of the 18th meeting of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pages 160–160. ACM, 2014.

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

[WHK17] Rene Winchenbach, Hendrik Hochstetter, and Andreas Kolb. Infinite continuous adaptivity for incompressible sph. *ACM Transactions on Graphics*, 2017.

[WHPS13] M. J. Woodhouse, A. J. Hogg, J. C. Philips, and R. S. Sparks. Interaction between volcanic plumes and wind during the 2010 Eyjafjallajökull eruption, Iceland. *J. Geophysical Research: Solid Earth*, 118:92–109, 2013.

[WKBB18] Marcel Weiler, Dan Koschier, Magnus Brand, and Jan Bender. A physically consistent implicit viscosity solver for sph fluids. *Computer Graphics Forum*, 37(2):145–155, 2018.

[WL98] JONATHAN J. WYLIE and JOHN R. LISTER. Stability of straining flow with surface cooling and temperature-dependent viscosity. *Journal of Fluid Mechanics*, 365:369–381, 1998.

[Woo88] A. W. Woods. The fluid dynamics and thermodynamics of eruption columns. *Bulletin of Volcanology*, 50(3):169–193, 1988.

[WP10] Steffen Weißmann and Ulrich Pinkall. Filament-Based Smoke with Vortex Shedding and Variational Reconnection. *ACM Transaction on Graphics*, 2010.

[WXCT19] Maximilian Werhahn, You Xie, Mengyu Chu, and Nils Thuerey. A Multi-Pass GAN for Fluid Flow Super-Resolution. *Symp. on Computer Animation*, 2019.

[XZWY20] Xiangyun Xiao, Yanqing Zhou, Hui Wang, and Xubo Yang. A novel cnn-based poisson solver for fluid simulation. *IEEE Transactions on Visualization and Computer Graphics*, 26(3):1454–1465, 2020.

[YLH+14] Chunqiang Yuan, Xiaohui Liang, Shiyu Hao, Yue Qi, and Qinping Zhao. Modelling cumulus cloud shape from a single image. *Computer Graphics Forum*, 33(6):288–297, 2014.

[YNBH11] Qizhi Yu, Fabrice Neyret, Eric Bruneton, and Nicolas Holzschuch. Lagrangian Texture Advection: Preserving both Spectrum and Velocity Field. *IEEE Transactions on Visualization and Computer Graphics*, 17(11):1612–1623, November 2011.

[ZB05] Yongning Zhu and Robert Bridson. Animating sand as a fluid. *ACM Trans. Graph.*, 24(3):965–972, jul 2005.

[ZBQC13] Cédric Zanni, Adrien Bernhardt, Maxime Quiblier, and Marie-Paule Cani. SCALe-invariant Integral Surfaces. *Computer Graphics Forum*, 32(8):219–232, December 2013.

[ZKL+17] Shenfan Zhang, Fanlong Kong, Chen Li, Changbo Wang, and Hong Qin. Hybrid modeling of multiphysical processes for particle-based volcano animation: Particle-based Volcano Animation. *Computer Animation and Virtual Worlds*, 28(3-4):e1758, May 2017.

ECOLE
DOCTORALE

Titre : Simulation visuelle efficace de phénomènes volcaniques

Mots clés : informatique graphique, animation, phénomènes naturels, modèles interactifs

Résumé : L'un des principaux défis en informatique graphique est de permettre la création de mondes virtuels vastes et réalistes. Pour les enrichir, il faut pouvoir offrir des outils pour créer des phénomènes naturels cohérents divers et variés facilement et efficacement. Ainsi, des effets visuellement plausibles mais contrôlables donnent plus de liberté pour la création de films 3D, de simulations et de jeux vidéo. Le but de cette thèse est de proposer de nouveaux modèles d'animation pour les éruptions volcaniques. Ces modèles sont pensés pour être utilisés par des artistes, posant certains objectifs à atteindre : être plausibles, c'est-à-dire avoir une ressemblance avec la réalité et une dynamique correcte ; être rapides, pour pouvoir être utilisés de manière interactive et dans les jeux vidéo ; et être contrôlables, en étant légers et simples à comprendre, pour que les artistes puissent les adapter à leurs besoins. Pour animer des éruptions explosives, nous proposons un modèle prenant en compte leur dynamique, simulant des panaches montant à haute altitude et des coulées pyroclastiques dévalant les flancs du volcan, selon les conditions initiales. Notre modèle comporte deux couches : une simulation Lagrangienne minimaliste représentant des tranches horizontales de panache interagissant avec l'air environnant ; et un modèle procédural renforçant l'animation des flux turbulents avec des détails multi-résolution. Nous étendons ce modèle en le combinant à un modèle atmosphérique où plusieurs couches horizontales représentent l'atmosphère, où nous simulons les phénomènes physiques menant à la formation des nuages. Ainsi, différents nuages peuvent être animés et influencés par la présence d'un panache. Enfin, les coulées de lave font partie des phénomènes naturels les plus complexes, puisque la lave est un fluide aux comportements multiples, passant par des états liquide, plastique et solide. L'aspect visuel de la lave est souvent hybride, avec des parties liquides sur lesquelles flottent des éléments plus froids, et des croûtes déformables qui se plissent et se déforment. Aucune méthode n'a pu gérer l'interaction entre l'aspect visuel de la lave et la coulée, ni la formation et le plissement de surfaces déformables. Plutôt que de s'intéresser à de la simulation pure, nous proposons un modèle à couches combinant une simulation Eulérienne de lave et une simulation géométrique de la surface de la coulée, permettant l'apparition de plis en fonction de la vitesse et de la température. Nous habillons la surface de la coulée avec des textures procédurales basées sur l'épaisseur de la croûte en respectant une cohérence temporelle.

Title : Efficient visual simulation of volcanic phenomena

Keywords : computer graphics, computer animation, natural phenomena, interactive models

Abstract : Offering tools to easily and efficiently create consistent natural phenomena is one of the main challenges in Computer Graphics, where visually plausible but controllable virtual effects are mandatory for 3D films, simulators and games. The goal of this PhD is to propose novel multi-scale models for animating volcanic eruptions. These models are meant to be used by artists, giving several goals to achieve: being plausible, which means having a geometric resemblance to reality, as well as having the correct temporal dynamic; being fast in order to be able to be used interactively while permitting an usage in video games; and finally being controllable, with light and easy to understand model, so that users can easily adapt them to their needs. To animate explosive eruptions, we propose a model that takes their unique dynamics into account, resulting into ascending plumes propagating upward and finally spreading side-way as well as pyroclastic flows spreading down the slopes of the volcano, depending on initial conditions. Our model combines two consistently coupled, simple sub-models: a minimalist Lagrangian simulation, used to represent dynamic horizontal slices of material ejected by the volcano and interacting with the surrounding air; and a procedural model that enhances the visual animation of the turbulent flow with multi-resolution details. We extend this model by combining it with an atmospheric model, where several horizontal layers represent the atmosphere and we simulate the physical phenomena leading to the formation of clouds. Thus, several types of clouds can be animated and interact with a plume. Lastly, lava flows are among the most complex targeted phenomena, since they involve fluids evolving to a variety of behaviors while cooling down, from liquid to plastic, and then to rigid states. The visual aspect of lava is often hybrid, with liquid parts carrying cooler elements, and deformable crust that folds and deforms. None of these methods was able to handle the interaction between the visual state of the lava and the underlying flow, nor the formation and folding of deformable surface sheets. Therefore, rather than tackling pure simulation, we use as a base an existing Eulerian simulation of lava flows and build a geometrical simulation of the surface of the flow, letting folds appear knowing the velocity and temperature of the flow. We texture the flow using time-consistent textures generated according to the thickness of the crust.

Institut Polytechnique de Paris
91120 Palaiseau, France