UNIVERSIDADE DA BEIRA INTERIOR
Engenharia

Realistic Simulation and Animation of Clouds using SkewT/LogP Diagrams

Rui Pedro Monteiro Amaro Duarte

Tese para obtenção do Grau de Doutor em
Engenharia Informática
(3° ciclo de estudos)

Orientador: Prof. Dr. Abel João Padrão Gomes

Covilhã, junho 2016

To Tomás

Acknowledgements

This endeavour has been a great voyage through peaks and dales, during which the contribution of some people was of major importance when the dales seemed almost impossible to leave and the peaks were far from reach. Their encouragement and comprehension were, per se, the changing point in many of the bad moments in this lonely journey.

Foremost, I want to express my endless gratitude and respect to my supervisor, Prof. Abel Gomes. In the course of this dissertation, Prof. Abel was someone who always believed in this work. In certain moments, he believed in it more than myself, always convincing me to continue and work harder that results would appear. During this thesis, I cannot forget of a question he always asked me: ”Is it done?”. This simple question always led me a step further to reach the final goal represented in this thesis. I am very grateful to him and wish that we can continue to work in the future.

I also want to thank Prof. José Morgado for the patience in the endless discussions about physics and weather phenomena. He always had a simple perspective for my complex problems. For him, my respect and admiration.

Finally, but not less important, I want to thank my parents that gave me the tools to build my future. This is something I will always remember with pride. But above all, I want to apologise to my wife Alexandra and to my son Tomás for not paying them enough attention when I was working on this thesis.

Resumo

Nuvens e clima são tópicos importantes em computação gráfica, nomeadamente na simulação e animação de fenómenos naturais. Tal deve-se ao facto de a simulação de fenómenos naturais —onde as nuvens estão incluídas— encontrar aplicações em filmes, jogos e simuladores de voo. Contudo, as técnicas existentes em computação gráfica apenas permitem representações de nuvens simplificadas, tornadas possíveis através de dinâmicas fictícias que imitam a realidade. O problema que este trabalho pretende abordar prende-se com a simulação de nuvens adequadas para utilização em ambientes virtuais, isto é, nuvens com dinâmica baseada em física que variam ao longo do tempo.

Em meteorologia é comum usar técnicas de simulação de nuvens baseadas em leis da física, contudo os sistemas atmosféricos de predição numérica são computacionalmente pesados e normalmente possuem maior precisão numérica do que o necessário em computação gráfica. Neste campo, torna-se necessário direcionar e ajustar as características físicas ou contornar a realidade de modo a atingir os objetivos artísticos, sendo um fator fundamental que faz com que a computação gráfica se distinga das ciências físicas. Contudo, simulações puramente baseadas em física geram soluções de acordo com regras predefinidas e tornam-se notoriamente difíceis de controlar.

De modo a enfrentar esses desafios desenvolvemos um novo método de simulação de nuvens baseado em física que possui a característica de ser computacionalmente leve e simula as propriedades dinâmicas relacionadas com a formação de nuvens. Este novo modelo evita resolver as equações físicas, ao apresentar uma solução explícita para essas equações através de diagramas termodinâmicos SkewT/LogP. O sistema incorpora dados reais de forma a simular os parâmetros necessários para a formação de nuvens. É especialmente adequado para a simulação de nuvens cumulus que se formam devido ao um processo convectivo. Esta abordagem permite não só reduzir os custos computacionais de métodos baseados em física, mas também fornece a possibilidade de controlar a forma e dinâmica de nuvens através do controlo dos níveis atmosféricos existentes no diagrama SkewT/LogP.

Nesta tese, abordámos também um outro desafio, que está relacionado com a simulação de nuvens orográficas. Do nosso conhecimento, esta é a primeira tentativa de simular a formação deste tipo de nuvens. A novidade deste método reside no fato de este tipo de nuvens serem não convectivas, o que se traduz no cálculo de outros níveis atmosféricos. Além disso, atendendo a que este tipo de nuvens se forma sobre montanhas, é também apresentado um algoritmo para determinar a influência da montanha sobre o movimento da nuvem.

Em resumo, esta dissertação apresenta um conjunto de algoritmos para a modelação e simulação de nuvens cumulus e orográficas, recorrendo a diagramas termodinâmicos SkewT/LogP pela primeira vez no campo da computação gráfica.

Palavras Chave

Simulação de Nuvens, Diagramas Termodinâmicos, Dados Atmosféricos

Abstract

Clouds and weather are important topics in computer graphics, in particular in the simulation and animation of natural phenomena. This is so because simulation of natural phenomena —where clouds are included— find applications in movies, games and flight simulators. However, existing techniques in computer graphics only offer the simplified cloud representations, possibly with fake dynamics that mimic the reality. The problem that this work addresses is how to find realistic simulation of cloud formation and evolution, that are suitable for virtual environments, i.e., clouds with physically-based dynamics over time.

It happens that techniques for cloud simulation are available within the area of meteorology, but numerical weather prediction systems based on physics laws are computationally expensive and provide more numerical accuracy than the required accuracy in computer graphics. In computer graphics, we often need to direct and adjust physical features, or even to bend the reality, to meet artistic goals, which is a key factor that makes computer graphics distinct from physical sciences. However, pure physically-based simulations evolve their solutions according to pre-set physics rules that are notoriously difficult to control.

In order to face these challenges we have developed a new lightweight physically-based cloud simulation scheme that simulates the dynamic properties of cloud formation. This new model avoids solving the physically-based equations typically used to simulate the formation of clouds by explicitly solving these equations using SkewT/LogP thermodynamic diagrams. The system incorporates a weather model that uses real data to simulate parameters related to cloud formation. This is specially suitable to the simulation of cumulus clouds, which result from a convective process. This approach not only reduces the computational costs of previous physically-based methods, but also provides a technique to control the shape and dynamics of clouds by handling the cloud levels in SkewT/LogP diagrams.

In this thesis, we have also tackled a new challenge, which is related to the simulation of orographic clouds. From our knowledge, this is the first attempt to simulate this type of cloud formation. The novelty in this method relates to the fact that these clouds are non-convective, so that different atmospheric levels have to be determined. Moreover, since orographic clouds form over mountains, we have also to determine the mountain influence in the cloud motion.

In summary, this thesis presents a set of algorithms for the modelling and simulation of cumulus and orographic clouds, taking advantage of the SkewT/LogP diagrams for the first time in the field of computer graphics.

Keywords

Cloud Simulation, Thermodynamic Diagrams, Sounding data

Contents

List of Figures

List of Tables

List of Acronyms

1D	One Dimension
2D	Two Dimensions
3D	Three Dimensions
AMS	American Meteorology Society
CA	Cellular Automata
CCL	Convective Condensation Level
CG	Computer Graphics
CFD	Computational Fluid Dynamics
CML	Coupled Map Lattice
CPU	Central Processing Unit
DALES	Dutch Atmospheric Large-eddy Simulation
DALR	Dry Adiabatic Lapse Rate
EL	Equilibrium Level
GALES	GPU-Atmospheric Large-eddy Simulation
GCM	General Circulation Model
GPGPU	General Purpose Computing Graphics Processing Unit
GPU	Graphics Processing Unit
LBM	Lattice-Boltzmann Method
LCL	Lifting Condensation Level
LES	Large-eddy Simulation
LFC	Level of Free Convection
MALR	Moist Adiabatic Lapse Rate
MLM	Mixed-Layer Model
PDE	Partial Differential Equation
UTC	Coordinated Universal Time (French: Temps Universel Coordonné)
VDB	Volumetric Dynamic Grids
WMO	World Meteorological Association

Chapter 1

Introduction

Clouds, water, fire, and smoke are natural phenomena, i.e., they are found in nature. However, because of their dynamics, they are not easily simulated and replicated on computer. In meteorology, the simulation procedures mainly aim at weather forecasting over the planet, not to visually reproduce clouds in a realistic manner on computer screen, as we try to do in computer graphics and applications like computer games, films or even flight simulators.

It happens that using computers to realistically simulate the dynamics of clouds is computationally expensive since their dynamics are regulated by physically-based equations. So, a major challenge of this doctoral work lies in to know whether it is feasible to reduce this computational burden of physically-based cloud systems in the scope of computer graphics in order to model, simulate, and realistically render clouds on computer screen in real-time.

1.1 Thesis Statement

The realistic simulation of the dynamics associated to cloud formation in the atmosphere requires the fulfilment of the following three requirements:

- Simulation of the convection process of clouds in the atmosphere.
- Realistic generation of cloud shapes that vary according to atmospheric conditions.
- Development of simulation methods for different types of clouds.

The simulation of the dynamics of clouds is well established by solving Navier-Stokes equations. However, these physically-based simulation methods are computationally very expensive and not suitable for real-time applications. This means that these physically accurate methods do not fulfil the real-time requirements of computer graphics applications. This explains why such applications search for the procedural generation of visually appealing clouds in real-time, discarding the underlying physics of their dynamics. That is, to achieve real-time rates, many methods in computer graphics for the simulation of cloud formation are procedural.

In respect to the first aforementioned requirement, the convection process (i.e., heating and lifting) of an air mass that gives rise to a cloud in the atmosphere is of major

importance in the definition of the cloud shape. While physically-based methods accurately represent the vertical displacement by solving the Navier-Stokes equations, procedural methods are based on heuristics that neglect such vertical displacement. In this thesis, we have developed a technique that *explicitly* solves the motion equation of clouds at interactive frame rates. Furthermore, we endow these clouds with horizontal dynamics that takes into account the effect of the wind.

As regards the second requirement, i.e., the realistic generation of clouds that vary according to atmospheric conditions, only a few methods available in the literature consider atmospheric data as an input to generate cloud shapes. We know from the meteorology science that different types of clouds relate to different atmospheric conditions, which may vary from one location to another over time, resulting in different cloud shapes. With this in mind, we have incorporated real weather data obtained from weather agencies to generate clouds that adapt to such atmospheric conditions. From our knowledge, this has never been done before.

In regard to the third requirement, i.e., simulation of different types of clouds, most methods available in the literature focus on the generation of cumulus clouds. In largely, this is so because they are not only the ones first seen by the viewer looking at the sky in a synthetic 3D scene, but also of their sharp outlines with flat bases and puffy domes which confer them a fluffy, beautiful appearance. However, other types of clouds form in the atmosphere under different conditions. In this thesis, we have developed a method that not only generates cumulus clouds, but also orographic clouds, i.e., clouds that form over mountains. As far as we know, this is the first attempt to generate orographic clouds, with the advantage that both cumulus and orographic clouds are generated through a similar method.

Taking into consideration the requirements mentioned above, we can formulate the *thesis statement*, as follows:

> *Is it physically possible to use real data to simulate the dynamics of cloud formation in real-time on CPU?*

As shown throughout this thesis, we intend to prove that this statement is true, provided that we use SkewT/LogP thermodynamic diagrams to solve the motion equation in real-time. Such thermodynamic diagrams are fed by real data acquired by weather agencies spread around the globe.

1.2 Research Questions

Considering the general problem underlying the simulation of cloud formation, as well as the aforementioned requirements and thesis statement, the main research questions that need to be answered are the following:

Is it possible to simulate the formation of physically-based clouds in real-time?

Realistic Simulation and Animation of Clouds using SkewT/LogP Diagrams

The simulation of clouds through physically-based methods based on Navier-Stokes equations is computationally expensive and not suitable for many computer graphics applications. In largely, this explains why most cloud simulation methods and techniques in computer graphics are procedural, i.e., they mimic the physical processes in real-time without using physics. Instead, procedural methods use and tune heuristic-based atmospheric parameters (e.g., cloud bottom and top, fake temperature profile, etc.) to achieve realistic results. On the contrary, we use SkewT/LogP thermodynamic diagrams to explicitly solve the motion equation in real-time on CPU without using any time-consuming integration technique. In fact, we use thermodynamic diagrams to determine the parameters required to "feed" the motion equation explicitly. Furthermore, this approach allows us to generate cumulus and orographic clouds in real-time, avoiding the heuristic process associated to procedural clouds, once they are all automatically determined.

Can real meteorological data be incorporated into a model to simulate the formation of clouds?

The use of meteorological data made available by weather agencies around the planet in the generation of cloud shapes in computer graphics has been done in the past [TBV02]. Essentially, one extracts a triangulated isosurface from meteorological data within an axis-aligned grid, from which one construct a cloud. However, this isosurface-based clouds are static, i.e., they are devoid of physics-based dynamics. In a way, we are talking about precomputed clouds that provide little freedom to generate other clouds using the same dataset. In this thesis we use a different type of datasets: atmospheric soundings obtained from weather balloons released in the atmosphere. Typically, in a certain region, these balloons are released by weather agencies every one, six, nine or twelve hours to obtain data about the vertical profile of the atmosphere. Using these data related to the vertical profile of the atmosphere, one can "feed" the motion equation with parameters obtained from atmospheric soundings with the help of thermodynamic diagrams.

Is it possible to develop a method that can generate various types of clouds?

Most methods available in the literature focus on the generation of cumulus clouds. The reason behind this fact is because cumulus clouds are those closer to us when we look at the sky, with the further particularity that they look beautiful and with well defined frontiers, and where light scatters and reflects within the cloud. In fact, for rendering purposes, these are the most suitable clouds to test the quality of proposed lighting and rendering methods. Even so, other types of clouds were simulated throughout the years, namely cirrus and stratus clouds [Gar85]. In this thesis, we develop a model that adapts to two of the most important families of clouds: cumulus clouds and orographic clouds. While the first family of clouds results from convection, the second is related to clouds that form over mountains. This is the first attempt to simulate clouds belonging to this second family in computer graphics.

How to control the final shape of generated clouds?

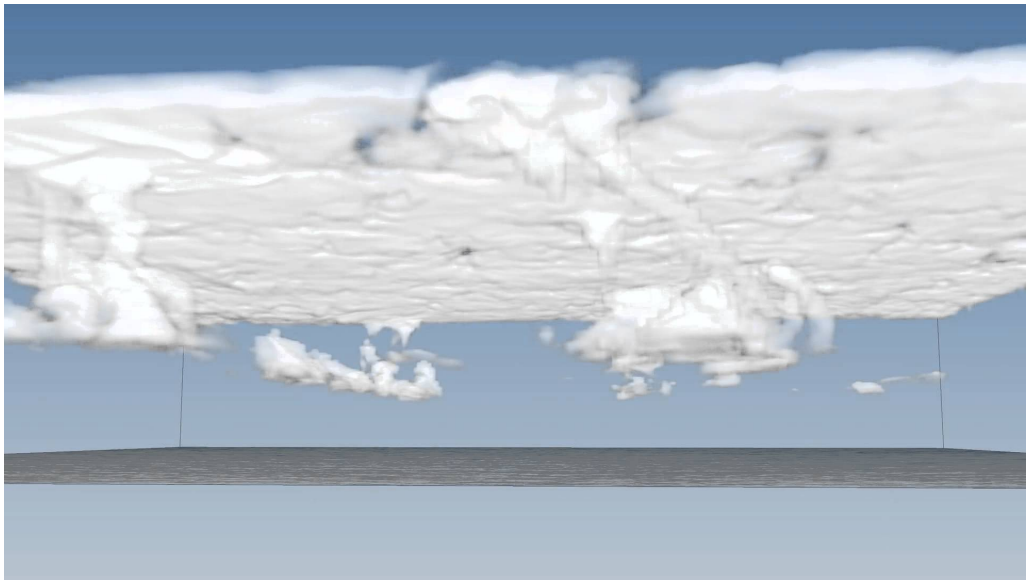
The control of a fluid in free motion (i.e., without boundaries) has been one of the major challenges in computer graphics. In our work, we control the bottom and the top of each cloud (i.e., its height) using a narrow or wider range of temperatures on the ground to trigger particles of an air mass in the atmosphere. In addition to the height of the cloud, it is also feasible to control the extent of the cloud using a smaller or a larger area of particle emitters on the ground. In order to have more control on the cloud shape, one can delineate the outline of a cloud, but that means to impose boundaries to a fluid. The study of a cloud as a fluid with boundaries has not been carried out in this thesis.

1.3 Research Context

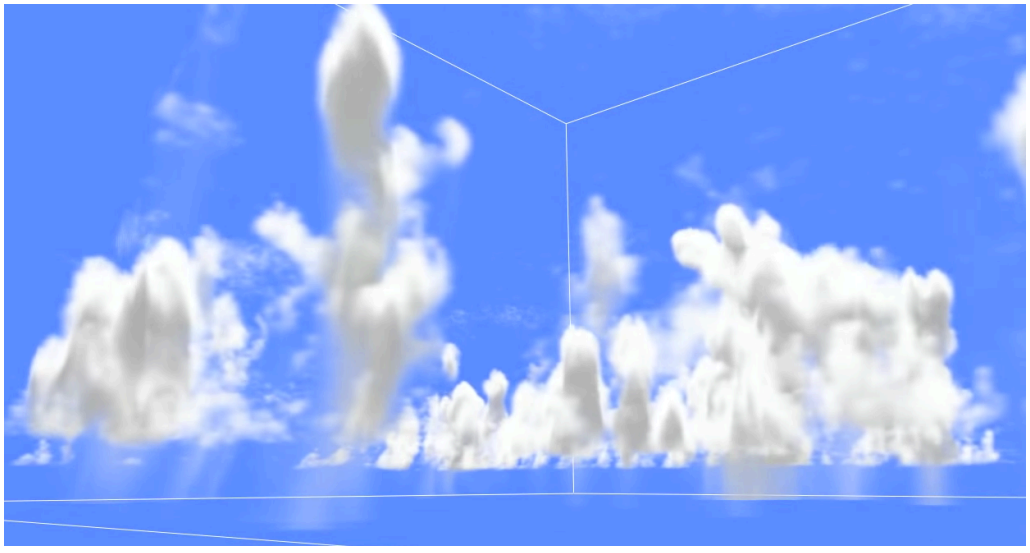
People have always been fascinated by clouds that form in the atmosphere. For the common person, it is amazing how these enormous and seemingly structures just float in the sky and appear and disappear out of nothing. It seems almost impossible that the air can hold such a mass and it does not fall into the ground (Figure 1.1). In some situations, these structures are of such size and magnitude that humans feel intimidated with their behaviour and impact on their daily routines, hence resulting a wide variety of emotions. In situations like the colourful clouds on sunset, a sensation of calmness and relaxation is felt when one observes them. This relation of astonishment and fear inspired artists throughout the centuries who incorporated this phenomenon into their paintings, books and stories.

Figure 1.1: A cumulus mediocris cloud formed over Baden-Wurttemberg, Germany. Marteng Vw

When scientific weather forecasting was not yet a routine, looking at the sky was the only way to understand weather in the near future and clouds were used as a tool to make such predictions. Nowadays, weather forecasting is important in many aspects of



(a) Three-dimensional visualization of a thin stratocumulus layer with cumulus updrafts appearing below as simulated with the Dutch Atmospheric LES (DALES) model at a resolution of 40 x 40 x 10 m^3. A video of part of the simulation can be found online (https://www.youtube.com/watch?v=vZpyVWye3S4)



(b) Deep convection simulation in a GPU-Atmospheric Large-Eddy Simulation, GALES. All calculation and visualization of this LES are performed using NVIDIA's CUDA. A video of part of the simulation can be found online (https://www.youtube.com/watch?v=mIUNRjBLIyQ)

Figure 1.2: Physically-based simulation models used in meteorology to simulate weather.

our daily lives. In particular, the meteorology science is important to study the impact of planet warming that results from the increase of the greenhouse gas concentration in the atmosphere. To predict such an impact, climate is being simulated using general circulation models (GCMs), mixed-layer models (MLMs) and large-eddy simulation (LES), for which the temperature increase is simulated. Here, the role of clouds is very important, since their presence in the atmosphere severely decreases the amount of energy from the sun that reaches the surface of the earth as they reflect a great part of this energy back to space. These modelling strategies of the atmosphere require a lot of computational power to generate simulations, once they rely on physics to numerically generate the atmospheric models. Examples of such simulators are the Dutch atmospheric large-eddy simulator (DALES) and the GPU-atmospheric large-eddy eimulation (GALES) [SGPJ12], where turbulent clouds are simulated using the GPU. Images of cloud formation and dynamics of these simulators are shown in Figure 1.2.

In computer graphics, a lot of research has been done in order to obtain photorealistic images of clouds since they are an indispensable element of any outdoor scene. However, most cloud models in computer graphics did not simulate the actual physical processes because they are extremely time-consuming. Instead, they follow a procedural approach, so that the resulting clouds depend on the artistic vein of computer animation professionals. In fact, artistic approaches require the tuning of several parameters to obtain a given shape for a cloud or to animate it somehow; for example, the dynamic of a cloud in the atmosphere may be faked with an oscillating motion regulated by a sinusoidal function. This trial-and-error process of tuning shape and dynamics of a cloud is also time consuming since many times artists are forced to start the process of cloud generation over and over, until a desired shape of a cloud is obtained.

Our motivation is thus to develop a lightweight physically-based method to simulate the formation of clouds from the ground up to their equilibrium level in the atmosphere, including their buoyant evolution that progressively gives place to their vanishment in the atmosphere. Our model should be able to explicitly model the physical processes involved in the advection of clouds in the atmosphere at interactive rates and render visually appealing clouds as is usual in computer graphics applications as movies, games, and flight simulators.

1.4 The Course of the Research Work

The research work underlying this thesis has aimed to simulate and render clouds on computer in real-time. Taking into consideration that a typical cumulus cloud takes about five to forty five minutes to form (http://usatoday30.usatoday.com/weather/resources/askjack/archives-clouds-precip.htm), we realize that it is possible to reproduce realistic clouds in real-time in the cloud time scale; this is so because this cloud time scale is much longer than frame rate.

Briefly speaking, the research work carried out in this doctoral thesis was developed into four stages. The *first stage* involved the study of the existing general-purpose models for clouds available in the literature, and how they could be useful to simulate the formation of clouds. Nevertheless, these models were mostly designed to cloud rendering, instead of cloud formation. Such methods fit in one of the following approaches: physically-based and procedural. Physically-based methods focus on solving Navier-Stokes equations to simulate the behaviour of a mass of air in the atmosphere, while the procedural methods mimic this behaviour heuristically. Physically-based methods are not able to generate clouds in real-time because of the computational costs associated to solving equations by integration. On the other hand, procedural methods achieve real-time frame rates, but they focus on the production of cloud shapes, not on its dynamics. This has led our research work to a point at which we asked ourselves whether it is feasible to combine the strengths of both approaches (i.e., physical realism and interactive frame rates) into some alternative approach.

The *second stage* of the work consisted in the building of such an alternative approach. Once the vertical profile of the atmosphere is a major issue to take into account in cloud simulation, we investigated on methods that realistically simulate the vertical motion of clouds in the atmosphere. Meteorologists use thermodynamic diagrams to forecast weather and determine the likelihood of cloud formation in the atmosphere under certain conditions. These diagrams are of major importance in our work, since they not only allow to understand the vertical profile of the atmosphere, but also to represent sounding data acquired by atmospheric balloons rising in the atmosphere. This allows us to calculate the top and bottom levels of a given cloud, as well as its extent in the atmosphere, and also other parameters that determine how the cloud evolves in the atmosphere. as well as other parameters that were heuristically determined in procedural methods. In fact, sounding data allow us to "feed" the motion equation associated to a given cloud so that such equation can be solved *explicitly*, This made it possible to simulate the lifecycle of cloud formation in the atmosphere at interactive rates.

The *third step* aimed at generating visually appealing clouds. To realistically render clouds, we have used Elementacular plugin for Maya software [Ale15]. This rendering technique first creates a 3D mesh from spheres representing particle positions and densities generated by our 3D simulator, being then a rasterization-based voxelization technique applied to the mesh. Three fields are subsequently across the voxelized domain where the cloud lies in: a distance field, a noise field based on fractals and a wispy field based on gridless advection. Finally, by applying a ray-marching technique, one gets the cloud illuminated in a realistic manner.

The final step of our research work aimed at generating other types of clouds. For that purpose, and after studying how other types of clouds could be simulated and rendered on screen, we noted that orographic clouds remained an unexplored issue. Orographic clouds, also called mountain clouds, account for the influence of the terrains on the

formation of clouds. With this in mind, we carried out a reworking on thermodynamic diagrams in order to adapt them to the forced lift motion of cloud against a given terrain.

1.5 Contributions of the Research

This thesis fits in the scope of computer graphics. Bearing in mind the challenges put forward in the previous sections, we can point out the following contributions to the advance of knowledge:

- *Thermodynamic diagrams.* Thermodynamic diagrams have been used for decades by meteorologists in weather forecasting. Their importance has to do with the fact that they allow us to determine the likelihood of cloud formations and, most commonly, to forecast thunderstorms. For that purpose, it is necessary to add air parcel temperature profiles acquired by an atmospheric balloons, which measure the variations of the temperature of one or more an air masses as they evolve in the atmosphere vertically. These diagrams are the core of our simulator, since they provide us a tool to automatically calculate parameters related to the atmospheric properties of cloud formation. As far as we know, thermodynamic diagrams have never been used in computer graphics to simulate the vertical motion of clouds in the atmosphere.
- *An explicit approach to the solution of the equation of motion of clouds in real-time.* A new physically-based cloud modelling method is put forward in this thesis. But, unlike other physically-based methods, we do not use numeric integration to solve the Navier-Stokes equations. Instead, in this research, the motion equation for clouds is explicitly solved by "feeding" it with information obtained from the SkewT/LogP thermodynamic diagram. This dramatically reduces the computational burden associated to solving Navier-Stokes equations, and allows us to generate clouds in real-time.
- *Incorporation of real meteorological data.* Many weather agencies make weather forecasting using atmospheric balloons, each one of which incorporates a radiosonde to transmit atmospheric data down to Earth. A radiosonde tied to a ballon is released in the atmosphere every one, six, nine or twelve hours (depending on the weather agency) that gathers data about the temperature profile of the atmosphere. These data are incorporated into our thermodynamic diagram in order to generate realistic cloud anywhere in a 3D synthetic scene, with the advantage that the synthetic clouds evolve over time.
- *Automatic generation of clouds that adapt to the atmospheric conditions.* A tool based on SkewT/LogP thermodynamic diagrams can be used by artists to control the overall shape of clouds. With such diagrams, artists are able to control the

top and bottom of a given cloud, by direct manipulation of temperature curves represented in the SkewT/LogP diagram interface of our simulator. It also allows the control of the cloud shape by changing the temperature and amount of particle emitters on the ground.

The major contribution of this thesis lies in the use of SkewT/LogP thermodynamic diagrams in computer graphics for the first time to simulate the formation of clouds over time. We show throughout this thesis that these diagrams apply not only to cumulus clouds, but also to other sorts of clouds; more specifically, orographic clouds. In fact, by incorporating terrains in the simulation, we are able to use thermodynamic diagrams to control the formation of clouds over mountainous regions. But, more importantly, one proposes an explicit approach to solve the motion equation of clouds in the atmosphere, which allows us to simulate the formation of clouds in real-time. Furthermore, this thesis presents an end-to-end integrated system for the simulation of cloud formation and its dynamics, though the rendering system is taken from Maya's Elementacular plugin [Ale15].

1.6 Applications

With the technological advances in hardware, game and film industries are continuously looking for new methods to develop the realism of their productions. The creation of convincing environments is of critical importance in game and movie industries because costumers' expectations grow with time. Recently, several movies and games have been released that account for clouds as an important factor in their interactions. Movies like *Partly Cloud* and *Good Dinosaur* and games like *World of Warplanes* and *Horizon: Zero Down* are some examples of movies and games that use realistic clouds in virtual scenes, as shown in Figure 1.3. Military and flight simulators are other applications where clouds also play an important role in realistic outdoor scenes.

1.7 Submitted Articles

The research work carried out in this thesis has given rise to the following paper submissions:

Rui P. Duarte, Francisco Morgado and Abel J.P. Gomes: Real-Time Simulation of Cumulus Clouds through SkewT/LogP Diagrams (submitted to *Computer Graphics Forum* after major revisions).

Abstract: The modelling, simulation, and realistic rendering of natural phenomena have been important goals in computer graphics for decades. Clouds, as a natural phenomenon, represent a real challenge because their birth, life, and death are amorphous and dynamic in nature. As a consequence, cloud simulation is computationally

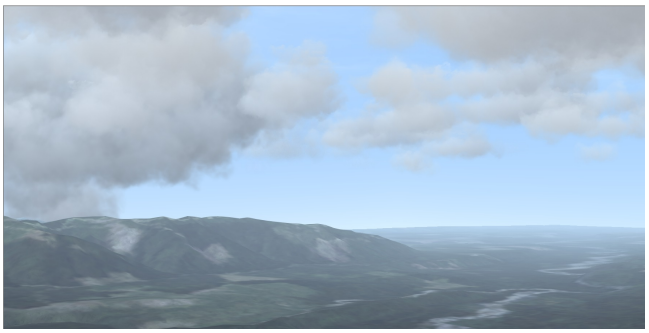
(a) *Partly Cloud* movie.

(b) *Good Dinosaur* movie.

(c) *World of Warplanes* game.

(d) *Horizon: Zero Down* game.

(e) *Microsoft Flight Simulator*.



(f) *Silverligning* plugin used in several applications.

Figure 1.3: Scenes of films and games where clouds play an important role in the realistic visualisation of the environment.

very expensive when using physically-based methods. In order to overcome this time performance problem, this paper presents a real-time method for the simulation of clouds using SkewT/LogP diagrams. These diagrams provide us with an explicit technique to simulate the formation of clouds from sounding data made public worldwide by weather agencies. To achieve real-time rates, our system is based on physics, but avoids solving differential equations. We have also built a visual tool for 2D SkewT/LogP diagrams that allows us to inspect, control and simulate the thermodynamic process of ascending clouds in the atmosphere, as well as a 3D synthetic environment where clouds are advected by buoyant forces. This lightweight technique enables the incorporation of our cloud simulator in systems tied to a number of industries, namely movies, virtual environments, and video games.

Rui P. Duarte, Francisco Morgado and Abel J.P. Gomes: Orographic Clouds (submitted to *ACM Transactions on Graphics*)

Realistic Simulation and Animation of Clouds using SkewT/LogP Diagrams

Abstract: In this paper we present an approach to the simulation of cloud formation over mountainous regions that accounts for the dynamics of cloud evolution and the influence of the mountain on the cloud shape. During the past three decades much work has been carried out in the simulation of cloud formation, due to their multiple applications in film animations, computer games, and flight simulators. However, at our best knowledge, this is the first attempt to simulate clouds that form under the influence of terrains, once most methods focus on the simulation of convective clouds. This work proposes a method for the simulation of orographic clouds using real atmospheric data, obtained from weather forecasting. In our approach, we use two-dimensional temperature profiles to model the model the behaviour of three-dimensional clouds based on particle systems.

Rui P. Duarte, Francisco Morgado and Abel J.P. Gomes: Techniques for Cloud Simulation in Computer Graphics: a Survey (submitted to *ACM Computing Surveys*).

Abstract: Most natural phenomena comprise three stages that have to do with life and death: formation, evolution and extinction. Clouds, as a natural phenomenon, account for these characteristics which are very difficult to reproduce on computer. Accordingly, simulation of cloud dynamics is amongst one of the greater challenges in computer graphics. In order to ascertain the current state of knowledge and research, an extensive review of the literature in cloud dynamics has been undertaken to identify the harness potential factors and gaps in implementation. This paper provides a critical analysis of methods found the literature, with a focus on the dynamics of cloud simulation. The leading idea aims at answering into the following question: "How is it possible to simulate clouds in computer graphics?". The analysis will be divided into two categories of cloud simulation methods: physically-based and procedural-based. After such a comprehensive analysis, future trends in simulating cloud in 3D synthetic scenes are put forward in the end.

1.8 Software and Hardware Tools

In terms of software development for the simulation of cloud formation, all the code was exclusively and fully designed and written by the author. The SkewT/LogP thermodynamic diagrams were implemented in C++ and OpenGL/GLUT/GLUI libraries. The 3D simulator was implemented also in C++ and OpenGL/GLUT/GLUI libraries. With the exception of Maya's Elementacular plugin [Ale15] used to render clouds, no other third-party software was used. To optimise the use of the plugin, several scripts were implemented in Python. Our implementation does not use parallel processing of any sort. Our results were obtained on a desktop computer equipped with and Intel Core i7 3.07 GHz running Windows operating system (v7), 24 GB of RAM, and an Nvidia Quadro 6000 6GB graphics card.

1.9 Thesis Organisation

This thesis was thought of as a regular dissertation. Therefore, it is not organised as a collection of published papers, though each core chapter has originated a paper submitted to a journal for publication. Summing up, this thesis is organised as follows:

Chapter 1. This chapter puts forward the thesis statement, the corresponding research questions, the scientific contributions to the advance of knowledge in computer graphics, in addition to the research context and the rationale that have led to elaborating this thesis.

Chapter 2. In this chapter, one reviews the most important methods related to cloud simulation, with a focus on the modelling of cloud shapes. Both procedural and physically-based methods are approached in a comprehensive analysis in order to identify their strengths and weaknesses in the simulation of cloud generation and formation in a synthetic atmosphere. For comparison purposes, representative images of different sorts of clouds are shown in Appendix A, Figures A.1 and A.2. Let us also to refer that an introduction to fluid dynamics in computer graphics, namely the Navier-Stokes equations, appears in Appendix B.

Chapter 3. This is the most prominent chapter of the thesis. It details our algorithm for real-time simulation of cumulus clouds using SkewT/LogP diagrams together with sounding data made public worldwide by weather agencies. As already mentioned above, this is the first cloud simulation method that takes advantage of atmospheric diagrams. In addition, this method solves the cloud motion equation in an explicit manner, using for that purpose the sounding data as input over time.

Chapter 4. In this chapter, we describe a cloud simulation method for orographic clouds, i.e., clouds that are formed over mountains. This method can be seen as an extension of the method detailed in Chapter 3. Nevertheless, several new concepts have to be considered in the formation of orographic clouds, namely windward and leeward winds. In other words, although we continue to use SkewT/LogP diagrams, new parameters have to be calculated to model and simulate the behaviour of clouds over hills and mountains.

Chapter 5. This chapter concludes the thesis, being also discussed the limitations of the proposed cloud simulation method. In the end, some hints for future work are put forward.

Summing up, the core of this thesis lies in the method presented in Chapter 3 provided that it allows us to simulate cumulus clouds in real-time. But, as shown in Chapter 4, this physically-based simulation method can be extended to orographic clouds and, eventually, to other sorts of clouds.

Chapter 2

Cloud Simulation in Computer Graphics: A Survey

Most natural phenomena comprise three distinct stages: birth (formation), life (evolution), and death (extinction). Clouds, as a natural phenomenon, account for this living lapse, which is very difficult to reproduce on computer. In fact, generation of clouds and simulation of their dynamics is amongst one of the greater challenges in computer graphics. In order to ascertain the current state of knowledge and research in this field, an extensive review of the literature in computational cloud generation and simulation has been undertaken to identify the harness potential factors and gaps in several algorithms. This chapter provides a critical analysis of these algorithms, with a focus on the simulation of cloud dynamics. It seeks to respond to the following question: "How is it possible to generate and simulate weather clouds on computer using computer graphics techniques?". This analysis covers the two main categories of cloud generation and simulation methods: physically-based methods and procedural methods. After this comprehensive analysis, we present our vision in respect to future research trends in weather cloud generation, simulation, and modelling.

2.1 Introduction

In our daily lives, we look at the sky and see a wide variety of weather formations that vary from small clouds to huge clouds, many of which are related to thunderstorms. The World Meteorological Organisation (WMO) provides a classification of clouds (Fig. 2.1) that serves as a guide for cloud classification [Mas57]. However, these weather clouds are difficult to reproduce in synthetic environments on computer. In fact, in addition to water, fire, and smoke, modelling and animating weather clouds is amongst the most challenging problems in computer graphics and animation.

In general terms, there are two main categories of methods to generate and simulate clouds on computer: physically-based methods and procedural methods. Physically-based methods build on solving the Navier-Stokes equations to model the dynamics of fluids —as it the case of clouds. Computational physics dynamics (CFD) is the field that focuses on the simulation of fluids on computer. At bringing CFD to computer graphics arena, we are allowed to generate realistic fluid animations. The downside of this pollination of CFD into computer graphics is that the current cutting-edge computer technology cannot process CFD in real time for huge amounts of fluids.

However, the objectives of CFD and computer graphics differ. While the first tries to accurately represent the physical properties and behaviour of the natural phenom-

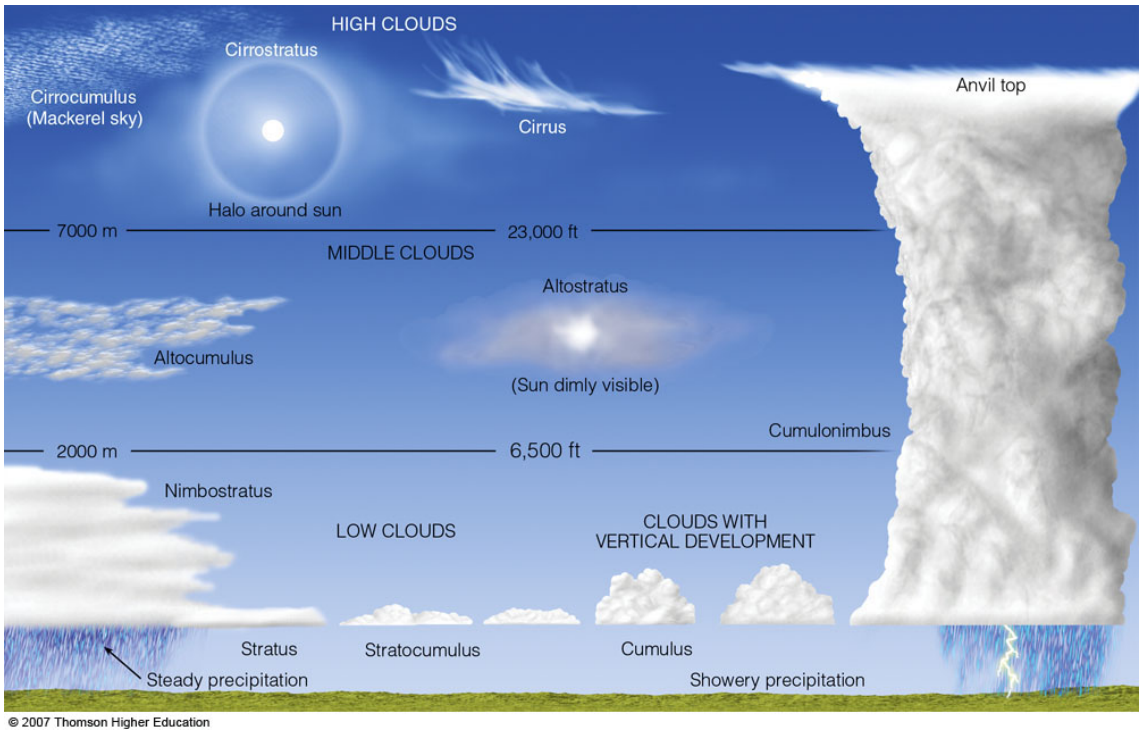


Figure 2.1: Cloud Classification. Clouds are classified in three levels of altitude: low, middle and high clouds depending on their formation process. ⓒⓘⓞ Thomson Higher Education

ena, the second neglects accuracy and focuses on visual appearance. This imbalance between these two approaches represents one of the major challenges in computer graphics nowadays. While CFD aims at reducing the computational time to generate their accurate models, computer graphics aims at reproducing visually realistic fluids, including clouds. In spite of the work that has been done in natural phenomena for the last decades, there is still a lot to do in order to fill the gap between the different purposes and applications in both fields, CFD and computer graphics.

The remainder of this chapter is organised as follows. This section introduces natural phenomena where we present an analysis of work carried out so far in surveying natural phenomena. A background on clouds is also presented. In section 2.2, methods used to simulate clouds based on physics are described and a general comparison is established. In section 2.3, we proceed as in section 2.2, for procedural methods. Section 2.4 refers to simulation methods based on images. Section 2.5 presents a discussion of the techniques and an insight on future trends ins presented. Section 2.6 concludes the paper, answering the question placed in the beginning of this chapter.

2.1.1 Background on Clouds

The atmosphere is a three dimensional fluid. The air moves horizontally and vertically creating weather phenomena that shape the properties of climate. In this context, clouds play an important role in the generation of realistic images of outdoor scenes.

The process of cloud development is referred as parcel theory [Nor38]. Conceptually,

parcel theory describes the vertical motion and changes of a parcel of air in the atmosphere. An air parcel is lifted by either buoyant or mechanical forces, causing it to cool off and expand as its pressure decreases. Eventually this air mass becomes saturated (relative humidity becomes 100%) and condenses into droplets, forming then a cloud. The saturation process occurs by the way of atmospheric mechanisms that cause the temperature of an air mass to be cooled to its dew point or frost point. This releases latent heat, which warms the air parcel and increases its buoyancy, making it rising further in the atmosphere. When the parcel reaches its maximum altitude, it begins to descend due to negative buoyancy, after which it ascends again due to positive buoyancy, so that it ends up oscillating around its equilibrium level.

The lifting of a parcel of air in the atmosphere is due to the following processes (or mechanisms):

- *Convectional lifting*. Convective clouds are a result from the heating of the air at the ground surface. If enough heating occurs, the mass of air becomes warmer and lighter than the air in the surrounding environment, and therefore it begins to rise, expand, and cool. When sufficient cooling has taken place, saturation occurs, forming clouds. The formation of convective clouds is described in Chapter 3.
- *Orographic uplift*. Orographic clouds result from air that is forced to rise by the action of windward winds that hit rising terrains, such as mountains, and leeward winds that force the air mass to descend. This results in a adiabatic expansion of the air at a rate of approximately 10° Celsius per 1,000 meters until saturation. In Chapter 4, we describe how orographic clouds are generated in 3D synthetic scenes.
- *Convergence or frontal lifting*. When two masses of air come together, the process of frontal lifting takes place. One of the air masses is usually warm and moist, while the other is cold and dry. The leading edge of the latter air mass acts as an inclined wall or front causing the moist warm air to be lifted.
- *Radiative cooling*. This process occurs when the sun is no longer supplying the ground and overlying air with energy derived from solar insolation (e.g., night). Instead, the surface of the Earth begins to lose energy in the form of long wave radiation, which causes the ground and air above it to cool. The clouds that result from this type of cooling take the form of surface fog.

These processes of cloud formation do not always act separately, being possible to have combinations of all four types. Clouds are classified using a Latin terminology to describe the appearance of clouds as seen by an viewer on the ground. This classification also takes into account the height of cloud base (Figure 2.1). For example, cloud names containing the prefix ”cirr-”, as in cirrus clouds, are located at high levels, while cloud names with the prefix ”alto-”, as in altostratus, are found at middle levels. The four principal categories of this classification system are: *cumulus*, *stratus*, *cirrus* and *nimbus*, which are briefly described in the following manner:

- *Cirrus*. Cirrus-category clouds are whitish and formed at high altitude, with white filaments essentially detached and wispy, in a way similarly to cotton-on-raw that looks very loose. These clouds essentially are non-convective, but occasionally they are subject to a small-scale high-altitude convection, what endow them a tufted or turreted appearance. Unlike lower-altitude clouds, these higher-altitude clouds do not produce precipitation.

- *Cumulus*. Cumulus-category clouds are convective and produce precipitation. A a result of a localised free-convective lift, cumuliform clouds may grow from the lower level to an upper level of the troposphere. In terms of shape, these clouds possess flat bases and puffy domed tops, being heaped, rolled, or rippled in the middle. Clouds of this category are the most commonly simulated in computer graphics.

- *Nimbus*. Nimbus-category clouds are rain clouds. In particular, cumulusnimbus-category clouds are highly convective and produce heavy rain and thunderstorms. These clouds are also associated to other severe weather conditions, including snow, hail, strong wind bursts and tornadoes.

- *Stratus*. Stratus-category clouds possess a flat sheet-like shape. The formation of these clouds occurs at any altitude in the troposphere since there is enough condensation as the result of non-convective lift of relatively stable air. Unlike free convective cumulus-category and cumulonimbus-category clouds, stratus-category clouds do not tend to grow upwards because of their lack of convective activity. This means that these clouds produce precipitation that is normally steady and widespread, although the intensity of the showers varies from light to heavy, what depends on the thickness of the stratiform layer.

Given all these types of clouds, typically cumulus clouds are the most common in computer graphics. This is due to their volumetric and fluffy appearance, which allows the validation of most methods developed during the last decades (80 % of the literature review in this paper consider cumulus clouds in their simulations).

2.1.2 Applications

Several industries like the film industry, games and military require the incorporation of natural phenomena in their systems to obtain realistic environments. In the film industry, some of these phenomena are not difficult to reproduce and can be easily filmed in real life. Others, like tsunamis or volcanos erupting lava and thunderstorms are not that common and need to be reproduced. Another application is film animation. As stated from one of the producers of *Shrek II*, the most difficult scene to animate was when Shrek drunk a cup of milk [EMF02]. In fact, realistic reproduction of these phenomena is not an easy task to carry out, but more and more required in the film

industry; realistic applications can be found in movies like *Avatar (2009)* and *Hobbit (2012)*.

Games are another industry where natural phenomena are present. In most games, natural scenarios are an important part of the gameplay, and the interaction with elements is a key feature in the game. For example, in *World of Warplanes* and *World of Battleships*, clouds, terrain and water are integrated in a natural scenery of airplane control and battle simulation. Very realistic water is present in *Bioshock* and *Crysis* and, more recently, amazing clouds were produced in *Horizon: Zero Down* [THS+15].

Other industries, like military, use simulators for their pilots (e.g., *Microsoft Flight Simulator*, *X-Plane* and *Virtual Pilot 3D*) where weather is one of the key elements to incorporate in a virtual scene. Clouds, rain and wind affect the perception of the pilot regarding the surrounding elements, which give him/her the feeling of presence and immersion as in real life.

2.1.3 A Taxonomy of Cloud Simulation Methods

A significant body of work has been published in the area of cloud simulation to meet the requirements of the aforementioned industries. Until now, there is no survey that integrates all the different perspectives in this field. We build on this to reach the goal of integrating the work carried out in the major fields of research in cloud formation and evolution, presenting then a comprehensive comparison between techniques in order to envisage future trends for further research.

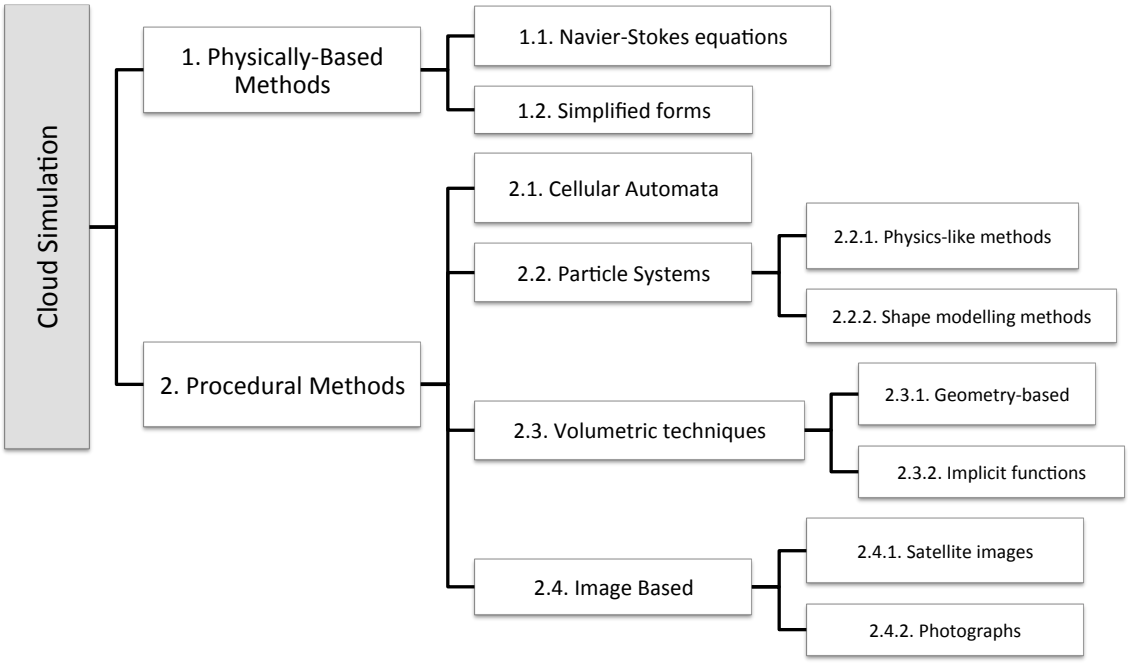


Figure 2.2: A taxonomy for cloud simulation approaches.

Let us then introduce a taxonomy of cloud simulation methods, as shown in Figure 2.2. In general, this taxonomy divides methods for cloud simulation into three major cat-

egories: physically-based methods, procedural methods, and image-based methods. *Physically-based methods* generate clouds by solving the Navier-Stokes equations. A few methods have been proposed in the literature to fully solve the Navier-Stokes equations (e.g., [MYND01, HBSL03, GSW07, DKNY08]). As expected, simplified forms of these methods have been developed in order to reduce the very time-consuming processing burden of solving Navier-Stokes differential equations using integration techniques [WPKK07, BWDY15].

Procedural methods follow two different approaches. In the first approach, one simulates the physics of clouds, but without solving the Navier-Stokes equations [Ney97]. In the second approach, there is no physics at all. This has led to the development of several techniques that better represent clouds rather than simulate their behaviour, namely cellular automata [DNO98, DKY+00], particle systems (e.g., [BN04, CSI09, REHL03]) and volumetric techniques (e.g.,[EMP+03, SSEH03, BNM+08, MMPZ12]).

Finally, we also consider *image-based methods* as a third category of methods for cloud generation. These methods follow a specific methodology that has been developed over the years (e.g., [DNYO98, DSY10, YLHY13, YLH+14]), which consists in extracting clouds from digital images of real life scenarios, independently of whether they are photography images or satellite images. These images are then handled in order to put them into synthetic environments like game worlds, virtual worlds or else. Similar to some procedural methods (e.g., cellular automata), image-based methods involve no physics either, and because of that they might be classified as procedural.

2.1.4 Other Surveys

In the computer graphics literature, we find various surveys that cover natural phenomena. However, apart the fact that they are mostly focused on fluids, they chiefly cover physically-based methods (see Table 2.1), i.e., they do not embrace procedural methods nor image-based methods.

Table 2.1: Surveys on the simulation and rendering of natural phenomena.

Reference	Type of Natural Phenomenon					
	Liquids	Fire	Explosions	Smoke	Clouds	Sand
Tan et al. [TY09]	●	●	●	•	•	●
Darles et al. [DCGG11]	●					
ZhaoHui et al. [ZZW11]		●				
Hufnagel et al. [HH12]					●	
Huang et al. [HGH15]				●		

● mainly focuses on; ● significantly focuses on; • focuses on; • slightly focuses on

As shown in Table 2.1, we have found five surveys on physically-based methods in the literature. The first survey is due to Tan et al. [TY09], which covers physically-based methods for fluids in general, with a particular focus on liquids, though methods to

generate fire, explosions, smoke, clouds, and sand were also approached.

Darles et al. [DCGG11] presented a very complete survey about simulation of ocean water, i.e., a particular case of a fluid. In addition to techniques to simulate the behaviour of ocean water, Darles et al. also introduced the major rendering techniques for ocean water.

On the other hand, ZhaoHui et al. [ZZW11] surveyed physically-based methods for fire simulation, while Huang et al. [HGH15] mainly focused on physically-based methods for smoke simulation.

In regard to weather clouds, the survey due to Tan et al. [TY09] seemingly was the first to include the topic of cloud simulation, yet it covers other fluids, with a particular predominance of liquids. However, Tan et al. were not concerned with the generation of visually realistic clouds on screen.

Instead of focusing on cloud simulation, Hufnagel et al. [HH12] surveyed the several cloud rendering and lighting methods existing in the literature. They also approached clouds with different types of volumes and boundary appearances. The physics behind these methods has to do with light scattering, self shadowing, and illumination of clouds; they were not interested in the physics of cloud generation as necessary in cloud simulation.

Finally, and unlike Hufnagel et al. [HH12], which focuses on cloud rendering, our survey mainly focus on cloud simulation, which is related to birth, life, and dead of clouds.

2.2 Physically-based methods

Most physically-based methods involve solving partial differential equations (PDEs), in particular the Navier-Stokes equations, which are as follows:

$$\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} + \frac{1}{\rho} \nabla p = \vec{g} + \nu \nabla \cdot \nabla \vec{u} \tag{2.1}$$

$$\nabla \cdot \vec{u} = 0 \tag{2.2}$$

where $\vec{u}$ stands for velocity, ρ is the density of the fluid, p denotes the pressure, g represents the gravity constant, and ν the viscosity of the fluid. The reader is referred to Appendix B for a detailed derivation of these equations. In order to solve the Navier-Stokes equations, one uses some sort of integration technique, what is very time-consuming in computer processing.

In a way, this accounts for why there are only a few physically-based methods to simulate the generation and formation of clouds and their dynamics in computer graphics.

In fact, in computer graphics, there is always a demand on fast simulations, largely due to emphasis on visual appearance and plausibility of clouds, rather than on accuracy.

2.2.1 Solving Navier-Stokes equations

The first attempt to simulate clouds in computer graphics using physically-based methods is the work of Kajiya and Von Herzen [KVH84]. Nevertheless they focus on a rendering method, the Navier-Stokes equations were solved to numerically model cumulus convection and generate three-dimensional optical density functions. Their model incorporates the equations of motion, continuity, condensation and evaporation, in conformity with the parcel theory. To account for water continuity, saturation mixing ratio was approximated using an exponential function of height. Despite the lack of processing power in that time, they were able to simulate the evolution of a cumulus cloud, as shown in Figure 2.3(a).

In 1999, Jos Stam [Sta99] introduced a method to solve the unstable physics equations with stable solutions, known as semi-Lagrangian method. The major contribution of this method is that, unlike many finite difference schemes, it is unconditionally stable, that is, the simulation is guaranteed not to diverge. These stability guarantees allowed this method to become the *de facto* standard advection algorithm in computer graphics. Basically, this method works as follows. In the advection step, a linear backtracking approach is applied across the grid. Instead of moving a point forward in space, one determines the source point in the opposite direction. Then the interpolated density value from that source is taken to the point. Shortly after the disclosure of Stam's method, several methods were developed based on this advection scheme [MYND01, OMK02, MDN02, HBSL03].

Overby et al. [OMK02] used Stam's semi-Lagrangian method to simulate cumulus and cirrus clouds at interactive rates (see Figure 2.3(b)). They improved on Kajiya's method by accounting with the variation in pressure within the atmosphere, and by neglecting viscous effects, which they assumed to be negligible in a standard atmospheric application. A different approach was used to model and solve the spatial effects of atmospheric fluid dynamics. The spatial grid is initialised with velocity, humidity, heat and condensation. In the source step, external velocities based on wind, vorticity confinement [FSJ01] and buoyancy are added to the system. In the simulation step, the fluid motion is handled by Stam's fluid solver. Their rendering is carried out in Skyworks software where they achieve real-time rendering.

Soon afterwards, and in order to take advantage of the graphics hardware (GPU), Harris et al. [HBSL03] achieved a significant reduction of the computational time in simulating clouds. Their model is based on a discrete form of previous techniques [FSJ01] to solve the same set of physically-based equations that model fluid flow, thermodynamics and water condensation and evaporation, as well as other forces. A staggered grid was used to represent the cloud shape, where the density is stored at the centre of each cell,

while the forces required for animation are located at the faces. With this approach, one obtains the low-definition clouds with blurry boundaries presented in Figure 2.3(e). To increase the realism of air currents, their method improves on the previous techniques [KVH84, OMK02] by accounting for negative buoyancy effects of the condensed water mass and the positive effects of water vapour. Note that Harris et al. considered that Overby et al.'s pointed out mainly two unrealistic effects: the computation of the expansion of rising air and the artificial momentum-conservation computation. For them, these effects are superfluous since the Navier-Stokes equations already account for them.

Using the semi-Lagrangian technique introduced by Stam, Miyazaki et al. [MYND01, MDN02] proposed a simplified numerical model to the formation of cumulus clouds (Figure 2.3(c)) based on the coupled map lattice (CML) method [YK97]. Here, Equation (2.1) and Equation (2.2) are approximated using the CML method, so that the following Navier-Stokes equation is obtained after dropping viscosity term:

$$\frac{\partial \vec{u}}{\partial t} = -\vec{u} \cdot \nabla \vec{u} + \eta \nabla \cdot \nabla \vec{u} + f \tag{2.3}$$

where η is a positive constant related to the rate of diffusion and f relates to the external forces. This technique is an extension of cellular automaton (CA) and the simulation space is divided into lattices. Each lattice has several state variables, which are real-value variables, so that they differ from those of CA approach, where they are discrete. Miyazaki et al.'s method accounts for phase transition and the effects of latent heat. The numerical dissipation in the advection step was addressed with the inclusion of the vorticity confinement technique [FSJ01]. They do not address the effects of the CA approximation on physical accuracy. Results from their method are rendered at 5 sec/frame on grid resolutions of $150 \times 120 \times 50$. Geist et al. [GRWS04] focused on cloud illumination using the Lattice-Boltzmann method (LBM). To generate the cloud density distribution, they combined the work of Miyazaki et al. [MDN02] with the work of Nagel et al. [NR92] (see Section 2.3.1). The first was used to define the macroscopic structure of clouds, while the second was used to treat the shape outputs as masks of humidity. In spite of the production of visually appealing clouds, (Figure 2.3(d)), they were unable to reach real-time rates. Later, they used the LBM technique to also account for the cloud density distribution [GSW07].

Physically-based methods were also used to simulate other types of clouds, namely volcanic clouds, tornadoes and earth scale clouds. In the modelling of volcanic clouds, little work has been carried out. Mizuno et al. [MDCN03] took two different approaches in simulating these types of clouds. First, they used a two fluids model where magma and entrained air are treated as two separate fluids. To account for the turbulent behaviour of the volcanic eruption cloud from a vent, they used the vorticity confinement technique. In their method, in a grid of 150 $\times$ 150 $\times$ 150 voxels, each simulation step takes about 7 seconds and 2 seconds per frame to render the final image. Then, [MDN04] simulated these clouds using the CML method. In this case, volcanic clouds are trans-

ported by atmospheric fluid and their density decreases due to the loss of pyroclasts. To account for this, the volcanic cloud density (ρ) equation is defined by:

$$\frac{\partial \rho}{\partial t} = -\vec{u} \cdot \nabla \rho - \kappa(z)\rho, \tag{2.4}$$

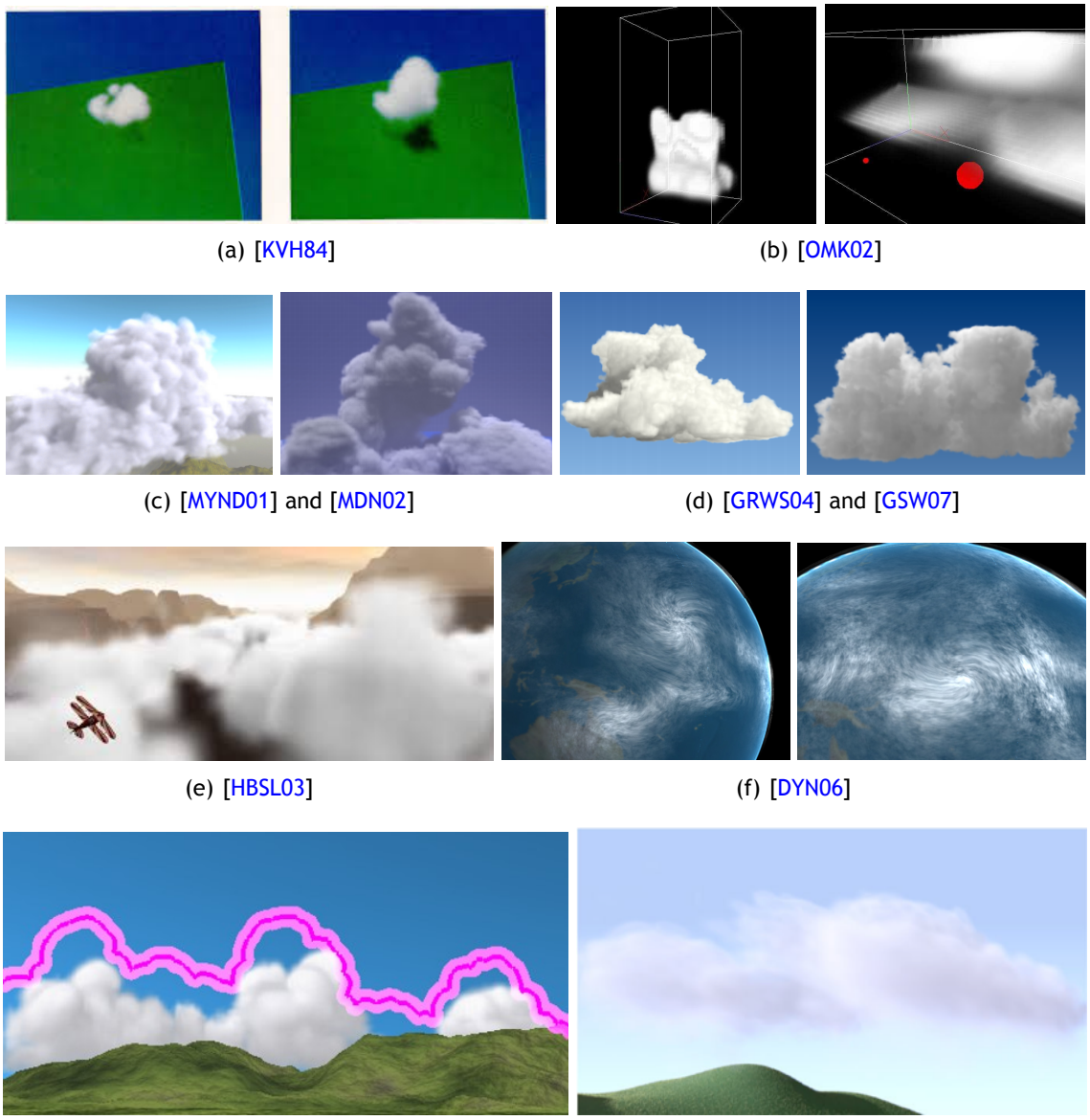
where $\kappa(z)$ is called the decreasing rate, which depends on two factors. First, near the vent, clouds consist of many large pyroclasts which implies a rapid decrease of ρ; this also implies that $\kappa(z)$ has to be set too large in this region. Second, at higher altitudes, clouds consist of small pyroclasts which implies a slow decrease of pyroclasts. In this case, $\kappa(z)$ has to be set small in this region. In their method, the shape of clouds is determined by the eruption magnitude, buoyancy, and decrease of volcanic cloud density. Later, Liu et al. [LWGP07, LWGP08] applied the two fluid model to the simulation of a tornado.

Dobashi et al. [DYN06] extended the work of Miyazaki et al. [MYND01] to simulate the motion of earth scale clouds (Figure 2.3(f)). Their method accounts for coriolis effect, friction between the atmosphere and the earth's surface and the formation of raindrops (this is used as a factor to determine cloud dissipation). Moreover, their method allows the control of clouds motion by specifying the atmosphere pressures on Earth's surface. The input for the method solver are four two-dimensional maps: atmospheric pressure and frictional coefficients on Earth's surface, surface temperature and evaporation rate. Using these maps, their method generates four 3-dimensional distributions: cloud density, density of raindrops, density of water vapour, and velocity. Since all the computations were done on CPU, they were limited to generate offline clouds.

2.2.2 Simplified forms

A few approaches have developed in order to reduce the computational burden of solving the Navier-Stokes equations. Wang et al. [WPKK07] decoupled the 3D into 1D vertical and 2D horizontal simulations using the similarity solution of turbulent plumes. In their decoupling cloud simulation, the user defines several parameters to establish the shape of the cloud, namely buoyancy effect, heat flux, and entrainment inflow. After this, the horizontal and the vertical directions of the evolution of the plumes (parcels) are treated separately. In the horizontal direction, they divide the 2D space into uniform grids and use Stam's fluid solver driven by differences in the virtual temperature that result from variations in the density of water vapour. In the vertical direction, they derive the properties of the parcel of air from the predefined parameters and using the similarity approach.

Barbosa et al. [BWDY15] took advantage of the works of Miyazaki et al. [MDN02] and Dobashi et al. [DKNY08] to apply them to particle systems. Instead of solving the Navier-Stokes, they simulated cloud formation based on position based fluids [MM13]. This technique is a particle-based method that uses position constraints to guarantee that the fluid density does not change. To smooth the scalar fields concerning temperature,



(a) [KVH84] (b) [OMK02]

(c) [MYND01] and [MDN02] (d) [GRWS04] and [GSW07]

(e) [HBSL03] (f) [DYN06]

(g) [DKNY08] (h) [BWDY15]

Figure 2.3: Examples of clouds generated with physically-based methods.

cloud density and vapour density, they used a method based on position based dynamics [MHHR07]. To shorten computation times, they solved their method on GPU. However, their results are not very realistic (Figure 2.3(h)), largely because of the rendering method they used.

2.2.3 Fluid Shape Control

As referred previously, one of the major issues with physically-based methods is the difficulty to control the behaviour of a fluid. For artists this is a major drawback, since they aim at specific shapes either for films or games, and not on non-controllable arbitrary shapes. Wang et al. [WPKK07] aimed at controlling the overall shape of clouds by defining overall distributions of vapour pressure. However, this is not easily captured

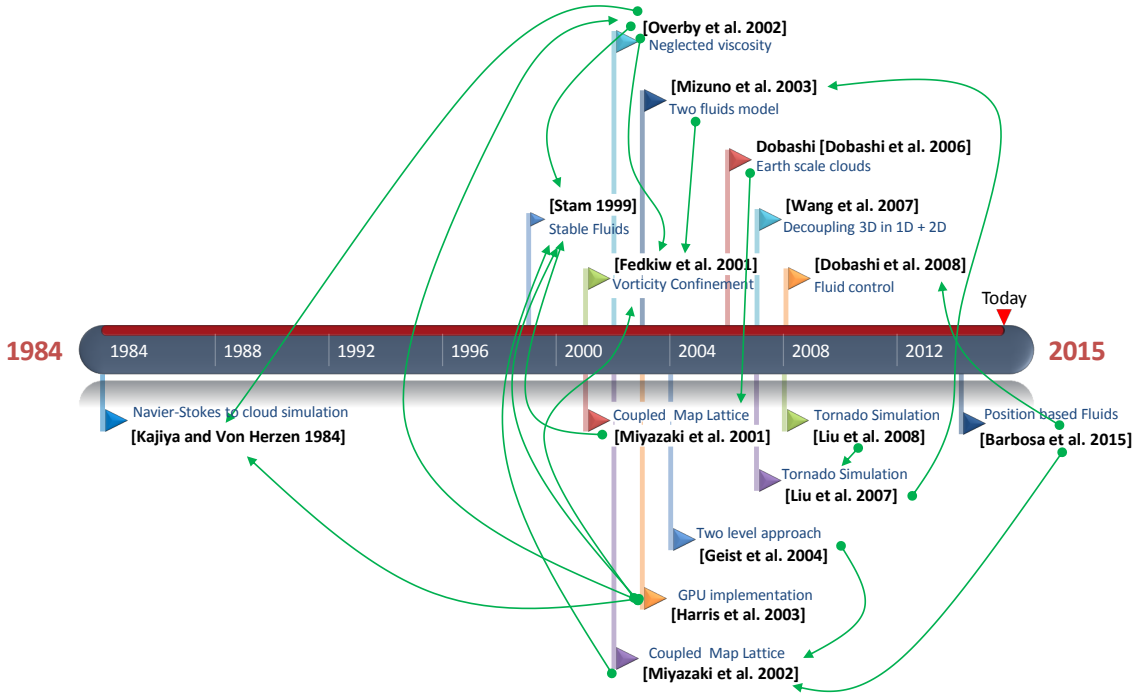


Figure 2.4: Timeline of physically-based methods. Green arrows represent relationships between methods. Each method has a small description of the major contribution in the field, either by improving on another method, or by an own contribution.

by most artists. A simpler attempt to overcome this problem, from the artist's point of view, was presented by Dobashi et al. [DKNY08] by allowing them to define the desired two-dimensional shape of the cloud to which the simulation would have to adjust parameters in order to generate clouds matching the desired shape. For the simulation, they use the method presented by Miyazaki et al. [MDN02]. To control the shape of the cloud, they adjust the amount of latent heat and water vapour within the simulation. As the cloud approaches the user defined silhouette, their control method determines the difference between the cloud position and the silhouette. This allows them to adjust the amount of latent heat required to increase/decrease the buoyancy of the cloud component. To prevent clouds from growing outside the desired shape, they determine the external forces due to a potential field. Barbosa et al. [BWDY15] also aimed at controlling the formation of clouds. Particle densities were used to detect "areas of interest". In areas of most interest, particles are split, while they are merged in areas with less interest, as in the work of Adams et al. [APKG07].

In Figure 2.4, a summary of the literature reviewed in this section is presented. To better visualise the relations established between works, green arrows are used to emphasise on those dependencies.

2.3 Procedural methods

Procedural methods in cloud simulation constitute an attempt to avoid the computational burden inherent to physically-based methods. Typically, they focus on two major concerns: simulation of cloud evolution and cloud modelling. In the previous section, we showed that physically-based methods typically focused on cloud formation and evolution since it is the core of solving the Navier-Stokes equations. However, this has the disadvantage that these simulations are computationally expensive and are not suited to real-time applications. There are three main classes of procedural methods, namely: cellular automata (CA), particle systems, and volumetric techniques. Although they are computationally inexpensive, they often require a trial and error approach to simulate the realistic dynamics of clouds.

2.3.1 Cellular Automata

Nagel and Raschke [NR92] proposed a cellular automaton as a discrete dynamical model consisting of a 3-dimensional regular grid (or lattice) of cells. At each cell, three state variables are assigned: *hum*, *cld* and *act* which denote vapour, clouds and state transition between vapour and clouds. Each variable is assigned to 0 or 1, and simulation is achieved by applying simple state transition rules in each time step t, as shown in Figure 2.5.

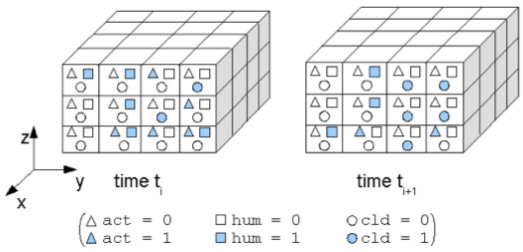


Figure 2.5: Simulation of the dynamic of clouds using CA.

A drawback of Nagel-Raschke method is that it does not account for extinction (since cld never returns to state 0, after reaching state 1) and realistic animation of cloud formation is difficult to achieve. To solve the extinction problem, Dobashi et al. [DNO98] introduced a new state variable *ext* and its transition rules. By supplying the three state variables at every frame with a random value from 0 to 1 (obeying a user-defined probability function), the variable *cld* changes its value in time, and extinction is achieved. However, this is still a problem, as we will see next. To solve the second problem, they used a continuous density distribution by using metaballs. To each metaball, a field function is defined and the continuous distribution is represented as a weighted sum of the field functions, and dynamic cloud evolution with extinction is achieved (first row of images in Figure 2.6).

Later, Dobashi et al. [DKY+00] improved on this method to overcome the problem of the repetition of cloud formation and its extinction in time (due to the transition of the *cld* variable from 0 to 1), which results in an unnatural animation. To account for the gradual extinction of clouds, their method incorporates an extinction probability p_{ext}

and a random number *rnd* such as $0 \leq rnd \leq 1$ in a cell whose *cld* is 1. *cld* is then changed to 0 if $rnd \leq p_{ext}$. With this extinction probability, the user can specify regions where extinction occurs more frequently. With this, the problem of cloud extinction is solved. However, it also introduces the converse problem. Once cloud is extinguished at a given cell, it never regenerates into that cell. To account for this, vapour (*hum*) and phase transition factors (*act*) are supplied to the system and two phase probabilities are used to set them randomly (similarly to the extinction probability). Wind is also introduced to account for cloud motion in one direction. To obtain a realistic representation of the cloud, a smoothing operation was carried out on the grid. But, this results in lack of detail, as can be seen in the second row of images in Figure 2.6. Their simulation can run at interactive rates that range from 10 to 30 secs/frame.

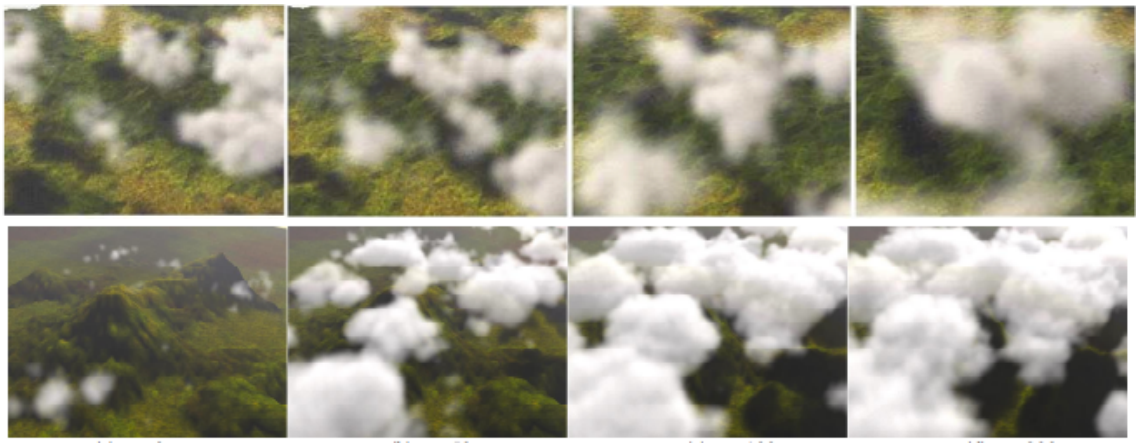


Figure 2.6: Clouds generated using Dobashi et al. cellular automata methods: [DNO98] (top row) and [DKY+00] (bottom row).

More recently, Shen et al. [SYL08] used Dobashi et al.'s method ([DKY+00]) in a marine search and rescue simulator. They improved the efficiency of the simulation method by splitting the model space into more smaller regions where each cloud is placed. They also improved the wind model to account for velocities in (x, y, z) directions, instead of a single direction, as in Dobashi et al.'s method. However, in visual terms, their clouds do not improve on previous methods. Upchurch and Semwal [US10] also considered wind in the 3D space. Instead of shifting cells within the CA in order to achieve wind motion, their method shifts ellipsoids that define the distribution of the probabilities. These values are calculated by pre-processing ellipsoids, storing probability values at each cell location. This approach allows them to have faster access to the probability values, although the pre-processing step is more time consuming when compared to the work of Dobashi et al. [DKY+00]. To generate stratus and cirrus clouds, they used a user-defined scalar dampening factor for each one of the probabilities. Setting the dampening factor high implies on puffy and full clouds, whereas lowering the dampening factor results in more sparse cloud structures. A timeline of these methods is shown in Figure 2.7.

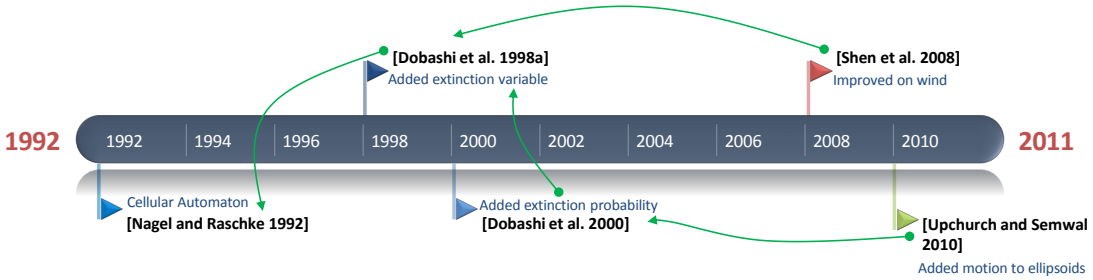


Figure 2.7: Timeline of cellular automata methods. Green arrows represent relationships between methods.

2.3.2 Particle Systems

In 1982, William Reeves was working on the film *Star Trek II: The Wrath of Khan*. During the film, a wall of fire ripples over the planet while it is being terraformed (Figure 2.8). This piece of film was the beginning of the use of the so-called particle systems in computer graphics: a class of "fuzzy" objects [Ree83].

Figure 2.8: Wall of fire from [Ree83].

A particle system is a technique that uses a large amount of particles to simulate complex phenomena, where each particle has a lifetime during which it may undergo several changes. The life of the particle begins when it is generated (or emitted by the system) and, at this point, several characteristics are assigned to the particle (velocity, size, rotation, colour, etc.). It is common to detect collision between particles and objects, though collisions between particles are rarely used, as they are computationally expensive and not visually relevant for most simulations.

In cloud simulation, particle systems have appeared in two distinct contexts in the literature. The first is in the simulation of cloud evolution over time, while the second is in the creation of manually-made clouds for artists. Since the aim of computer graphics is to focus on the visual appearance of clouds, most works carried out are based on the use of manually-made clouds to create realistic images by means of rendering algorithms.

Due to the simplicity of particle systems, they were rapidly adopted in cloud modelling. In the atmosphere, particles within the clouds evolve in the atmosphere according to the parcel theory, as explained in Section 2.1.1. Generally, the particle lifecycle is given by the three main attributes: *generation*, *evolution* and *dissipation*. Generation occurs when the temperature of each particle on the ground reaches the convective temperature. This makes them unstable to a point that they start rising in the atmosphere as dry particles. These particles evolve in the atmosphere while equilibrium is

not reached. Temperature trajectories are defined by the dry and moist lift motion, that is, motion before and after condensation. This defines their buoyancy force that allows the vertical motion of particles (cf. Equation B.28 in Appendix B.3). During this process, particles may dissipate (death) due to two major mechanisms: the mixing ratio of the air particle is lower than a specific threshold, or due to small scale changes in air moisture in the cloud shape.

In these methods, it is very common to have particle positions generated from simulation methods transformed into volumetric surfaces for rendering purposes [BN04, CSI09, Yus14]. This transformation is very useful for rendering methods that take advantage of GPUs. In this section, we will focus on relevant work carried out in stochastic cloud simulation and evolution using particle systems, as indicated in Figure 2.9.

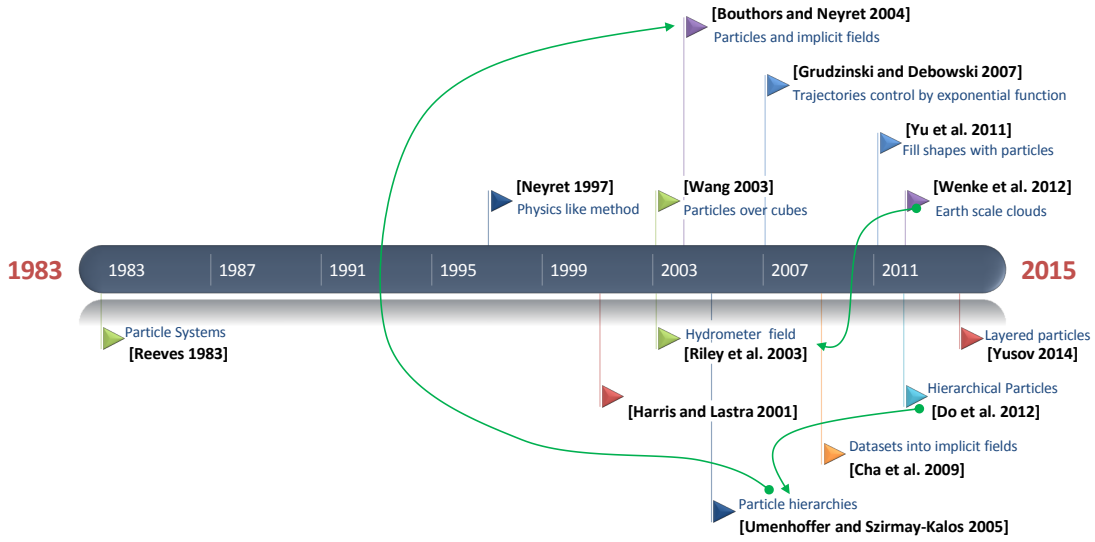


Figure 2.9: Timeline of methods based on particle systems to simulate clouds. Green arrows represent relationships between methods.

2.3.2.1 Physics-like Methods

This class of methods aims at simulating cloud formation by procedurally mimicking the physics dynamics. Neyret [Ney97] presented a method for the simulation of convective clouds based on qualitative formulations of particles within a cloud. In their method, parcels of air were simulated as ”bubbles” risen in turrets that formed the surface of the cloud. To mimic the visual appearance and growth of clouds, they used a macroscopic approach that characterises many specific structures appearing in cloud simulation, such as Raileigh-Taylor instability, bubble thermodynamic evolution and Benard cell. Despite the promising simulation results, this method has some disadvantages, namely the altitude at which a cloud forms, the environmental lapse rate and the air parcel lapse rate are all constant.

In Grudzinsky and Debowski [GD07], particles are generated according to a 2D matrix function (a generalisation of a probability density function) that covers the simulation

area. This function controls the shape of generated clouds since it depends on temperature. Low temperatures will "flatten" the function and generate stratus clouds, whereas a function with significant variations in temperature generates cumulus clouds. To account for the vertical trajectory of rising particles, they used an exponential trajectory function

$$z = h \,.\, \left(1 - e^{-\frac{d}{c}}\right), \tag{2.5}$$

where z is the particle's ceiling, h is its maximum ceiling, d is the distance between current particle's position and the point where it was generated and c is a constant. The major disadvantage of thismethod is that it is not simple for artists to generate cloud shapes from the exponential function since each particle will be assigned a trajectory. Also, this trajectory function does not account for downdrafts or buoyancy effect; as a consequence, it is not suitable to simulate cloud evolution over time.

2.3.2.2 Shape Modelling Methods

Other types of procedural methods focus on the definition of the cloud shape. The focus is not to reproduce the physical behaviour, but to generate shapes of clouds that can easily be used by artists. For example, Harris and Lastra [HL01, HL02] used particle systems to generate static cloud shapes for rendering. Basically, clouds are obtained from filling a volume with particles or by using an interactive tool that allows designers to create clouds. However, their method does no account for dynamic cloud simulation.

Wang [Wan03, Wan04] defined the overall shape of a cloud by creating and placing a series of boxes, as shown in Figure 2.10(a). With this manual modelling technique, artists are able to control and model the shape of clouds, including cumulus, stratus, and cumulonimbus clouds. Cirrus clouds, given their lack of volumetry are represented with flat rectangles. This method accounts for the formation and dissipation of clouds. Cloud formation is controlled by adjusting the transparency level of particle sprites, which depends on the position of the particle within the cloud. When the cloud is forming, visible particles with higher transparency value are those whose centre is within half of the cloud radius from the cloud centre. When they reach a threshold opacity, visible particles are those whose centre is over half of the cloud radius from the cloud centre. Dissipation is simulated by reversing this process.

Hybrid methods that use particle systems coupled with other techniques have been developed to take advantage of the benefits that each technique provides to cloud rendering [BN04, CSI09, DBLR12]. Bouthors and Neyret [BN04] modelled cumulus clouds by combining particles with implicit fields (Figure 2.10(c)). Their cloud modelling method is based on a hierarchy of particle levels. Each level l consists of several particles that define a surface S. Particles at level $l+1$ are generated over the surface S_l and parameters like location, radius, flattening and blending are generated to each particle. This results in surface S_{l+1}. The process terminates when the user achieves a desired cloud

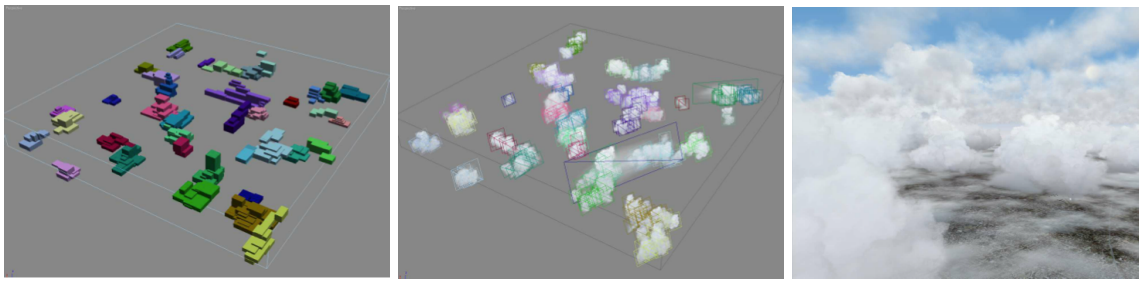
shape, that is given by the last surface obtained. This method is simple for artists to model cloud shapes, because each level only requires adding more a few particles. In a way, this process mimics the cloud evolution in the atmosphere.

Following a similar approach, Umenhoffer and Szirmay-Kalos [USK05] created the shape of clouds using the self-defined concept of particle hierarchies. Basically, clouds are built as blocks of particles, where each block represents many particles, as shown in Figure 2.10(d). The definition of particle hierarchies comes from the fact that a cloud can be built of smaller similar blocks. Similarly to Harris and Lastra [HL01], their focus is a rendering algorithm, and they improve on the efficiency of Harris-Lastra's method by considering that blocks are used to render depth impostors, instead of rendering billboards. In a non-hierarchical method, if a system has N particles, they build a block of b particles and instance it N/b times. In contrast to the N calculations in a non-hierarchical method, their approach requires only $b + N/b$ calculations.

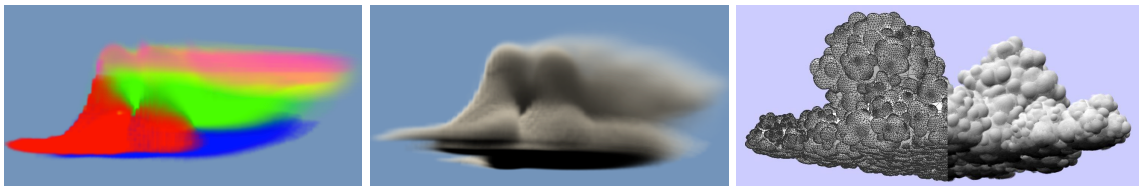
Later, Do et al. [DBLR12] used hierarchical particles to generate three types of clouds: cumulus, stratus and cirrus. In their method, the user generates large particles (seed particles) to define the overall shape of the clouds where each particle has a set of attributes, namely: radius, centre position and repulsion radius. Next, child particles are generated using a simplified version of Levet's particle sample method [LGS06]. Since clouds have different shapes, they created a set of heuristics to control the radius of particles generated for each cloud type. For cumulus clouds, their strategy is to reduce the repulsion radius in order to have a greater concentration of particles within an area (Figure 2.10(h)). For stratus clouds, the process is opposite: increase the repulsion radius to have particles more distributed in an area. Their method seems to be adequate to represent cumulus clouds, but cirrus and stratus clouds are difficult to model with particles due to their flat appearance.

To take advantage of GPU processing to draw triangles, Cha et al. [CSI09] transformed large data sets of particles into grid volumetric density fields (Figure 2.10(f)). A concern in obtaining these density fields relates to the accuracy of the smoothing length of neighbouring particles involved. Typically, using fewer particles results in low accuracy, while too many particles may improve the accuracy of microscale properties, but with a further computational cost. So, to get a consistent level of accuracy, the smoothing length is chosen to reflect the local density of particles. Their smoothing method works as follows. First, slices containing particles that may affect grid points are generated. Second, these slices are processed on GPU, so that particles are orthogonally projected onto the image plane by drawing squares centred at their respective positions in the volume space. Finally, after processing all the slices, they are assembled into a volumetric density field.

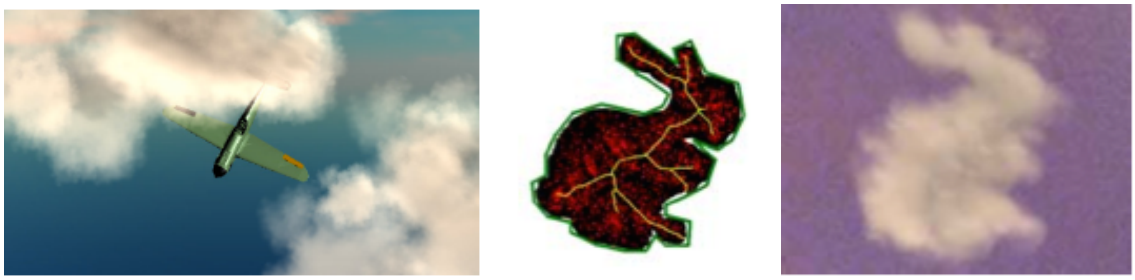
Yu et al. [YW+11] presented a technique to model 3D clouds based on two primitives: a 2D target shape or a 3D model. The 2D shape is transformed into a polygon structure by tracing the shape contour which is uniformly sampled with particles. To achieve a volumetric appearance they stretch the 2D particles to the 3D space. Because uniform



(a) [Wan03]



(b) [REHL03] (c) [BN04]



(d) [USK05] (e) [YW+11]

(f) [CSI09] (g) [WSYM12]

(h) [DBLR12] (i) [Yus14]

Figure 2.10: Examples of clouds generated with particle systems.

sampling makes the details of the target shape disappear, they adjust the distribution in order to have a tighter concentration of particles in the centre and sparser around the boundaries. To achieve this, they use a heuristic technique to control the shifting distance of a particle. First, the medial axis of the 2D shape is produced. Then, each particle is placed according to the medial axis until it fits the 2D shape. In the 3D model, they use a similar approach. To produce particles that are tighter in the centre and sparser around the edge, they combined voxels and skeletonization techniques [SK99, CSYB05] to divide the 3D model in tetrahedrons from the points on the medial axis, as shown in Figure 2.10(e).

More recently, Yusov [Yus14] generated particle-based cumulus clouds on GPU, with a camera-centred grid structure to account for expansive view distances. Rectangular rings are generated around the camera with the characteristic that cells in the middle ring are half the size of cells in outer rings. Then, layers containing particles are generated in each cell. Noise controls the number of layers, particle size and density. The size of particles varies depending the layer they are generated. For higher layers, particles are smaller than those generated in lower layers. With this approach, they were able to generate cumulus clouds, as shown in Figure 2.10(i).

2.3.2.3 Weather Data

In the field of cloud visualisation from weather data using particles, Riley et al. [REHL03] proposed a technique to visualise weather particle fields based on particle properties and represented on a grid (Figure 2.10(b)). To accurately visualise these particles, they convert the input data fields generated from physically-based simulation into particle concentrations. This is achieved by using the concept of hydrometer fields that can be of several types: cloud, ice, rain, snow and soft hail. To each field, a colour is assigned and is determined how particles within each field interact with light. They also used procedural noise to give detail to the simulated data. Later, Wenke et al. [WSYM12] improved on this method to generate earth scale clouds (Figure 2.10(g)) where each grid point of the imported data is treated as a particle. Since the weather forecast data contains a large number of grids, they reduce the complexity of the system by only considering particles at each grid point that possess an extinction coefficient higher than a threshold of 10^{-6}.

2.3.3 Volumetric Techniques

Due to the volumetric nature of clouds, volumetric modelling is vital for realistic simulation of this phenomenon in computer graphics. The use of summative implicit functions [GVJ+09] as a modelling primitive allows the user to take advantage of smooth blending of primitives and a more natural animation control. This technique has been combined with other techniques to generate the shape of clouds, namely particle systems (cf.

Section 2.3.2) and Image based modelling (cf. Section 2.4). In the first case, the simulation of a cloud shape is carried out using particles and, in the rendering step, these particles are combined to be interpreted as an isosurface. In the second case, the information extracted from images is treated as a isosurface to account for the volumetric appearance. However, since in those methods the shape of clouds is not directly defined by volumetric implicits, we refer the reader to Table 2.2 for a general view of methods that are coupled with volumetric implicit functions.

2.3.3.1 Ellipsoids and Metaballs

In this section, we focus on volumetric implicit methods as a basis to define the overall shape of clouds. In 1985, Gardner [Gar85] generated volumetric clouds (Figure 2.11(a)) using textured clustered ellipsoids, a mathematical texturing function (based on Fourier series) and translucence thresholds to model the cloud detail. To model different types of clouds, they use round ellipsoids for cumulus clouds and flat ellipsoids for stratus clouds. Their method also accounts for vertical and horizontal cloud development. Later, Elinas and Stürzlinger [ES00] improved on this method and, instead of using Fourier synthesis for the texture, they used and implemented Perlin noise on GPU, achieving so real-time frame rates.

A different approach was taken by Nishita et al. [NDN96], who used volumetric implicit functions to model clouds where its density distribution is defined using metaballs. First devised by Blinn [Bli82], these metaballs are arranged to form the basic shape of the cloud. Following the fractal method [Vos85], other metaballs are placed recursively over the surface of the cloud to generate details inherent to clouds. When a shape of cloud is obtained, the marching cubes technique is applied to the metaballs to generate the isosurface of the cloud. However, due to the lack of noise addition and their grid approach, the visual results suffer from the low level of detail associated to this type of methods.

Ebert et al. [EMP+03, Ebe04] presented a two-level approach to model clouds based on implicit functions. The cloud modelling macrostructure level is based on implicit geometric primitives [Gar85, ES00] defined and placed by the user. At the microstructure level they differ from Nishita et al. [NDN96]. Instead of using fractals, the macrostructure of the cloud is perturbed using Perlin noise and turbulence functions [Per85] (latter improved by Green [Gre05]) to account for small scale details of clouds. This improved the visual results of clouds when compared to Nishita et al.'s work. By changing parameters at the macrostructure level, different types of clouds can be created using his method (Figure 2.11(b)). However, they were unable to achieve real-time frame rates. Kniss et al. [KPH+03] used Ebert et al.'s method to define the cloud's macrostructure. However, for the microstructure of the cloud, they used two approaches. The first is based on Ebert et al.'s work, where a noise is added to modify the the optical properties of the volume. The second uses Peachey's noise simulation technique [EMP+03],

as illustrated in Figure 2.11(c).

Building on Ebert's work, Schpok et al. [SSEH03] implemented this two-tier approach on GPU. The macrostructure method involved volumetric implicits that could be controlled by the user. At the microstructure level, they used 3-dimensional textures generated with periodic noise data. This texture is subtracted from the macrostructure implicit volume. Figure 2.11(d) illustrates the described method. By using a less refined illumination model, they were able to reach interactive frames, between 5 and 30 frames/sec. Their method was latter used in the movie entitled *Valiant*. Later, Hufnagel et al. [HHS07] improved further on the metaball technique proposed by Dobashi et al. [DNYO98] (Section 2.4.1) with special attention to large scale visualisation of clouds from weather data.

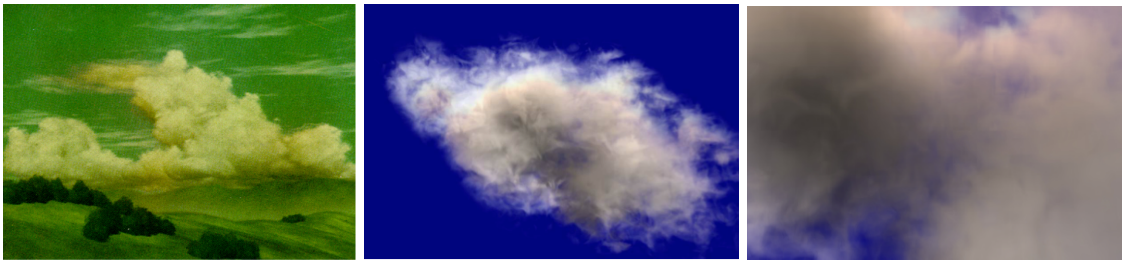
By using ellipsoids or metaballs and by disposing them in a fractal-like shape, most authors achieve the general outline of clouds. In most of these works, the application of noise over the metaballs greatly increases the level of detail in the shape of the clouds, resulting on visually appealing shapes.

2.3.3.2 Grid-based methods

Volumetric modelling of clouds with grids or meshes is also common as a support for rendering techniques. Trembilski et al. [TBV02] generated surface clouds from data produced by routine meteorological weather simulation. The cloud isosurface is evaluated, sampled and displayed using the marching cubes algorithm. To account for details in the cloud shape, they built upon the technique presented by Neyret [Ney97] (see Section 2.3.2) to account for the deformation of the refined surface. In their method, spheres are placed into the cloud structure at a random position. Then, the midpoint of the sphere is shifted up according to a pseudo-random Fourier wave function [Gar85]. Note that this is a time consuming process, but the result is a smoothed geometry (Figure 2.12(a)).

Riley et al. [RSK+06] developed a tool that allows the 3-dimensional visualisation of weather data measured using radars, using nonuniform grids. This data is imported into system and a transfer function is used for the mapping onto a nonuniform grid. Then, a 3-dimensional texture is generated based on a per-pixel transformation that determines if the point lies within the data domain. If so, a data sampling theatre coordinate is determined. Based on this technique, Song et al. [SYS+06] integrated the visualisation of one, two and three dimensional atmospheric datasets using grid structures (Figure 2.12(d)).

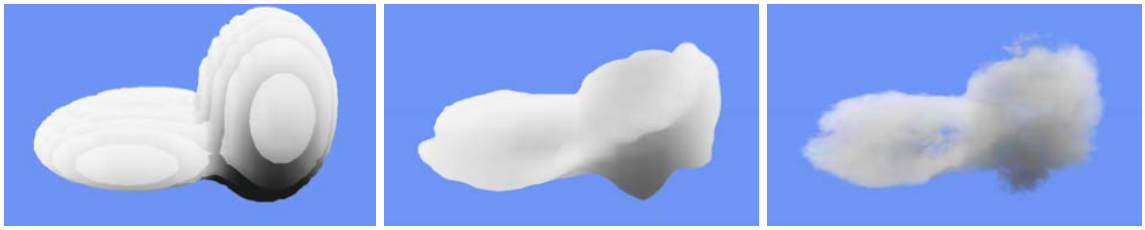
Bouthors et al. proposed a method for light scattering in stratiform [BNL06] and cumulus [BNM+08] clouds, as shown in Figure 2.12(b) and Figure 2.12(c), respectively. To model stratiform clouds, first, a mesh is used to describe the outer cloud boundaries at a low resolution, and second, a procedural volumetric hypertexture adds details to the boundary up to a certain depth (using a height field) to a point that finally the core of the



(a) [Gar85]

(b) Cumulus (left) and stratus (right) clouds generated using [EMP+03] method.

(c) [KPH+03] method: volumetric spheres defined by the user (left) and final images with different noise perturbations.



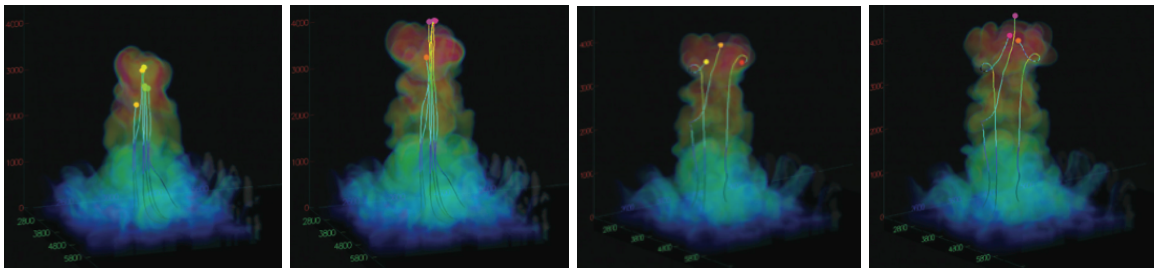
(d) [SSEH03] method (from left to right): Volumetric spheres defined by the user, noise added and final image.

Figure 2.11: Examples of clouds generated with volumetric primitives such as ellipsoids or metaballs.

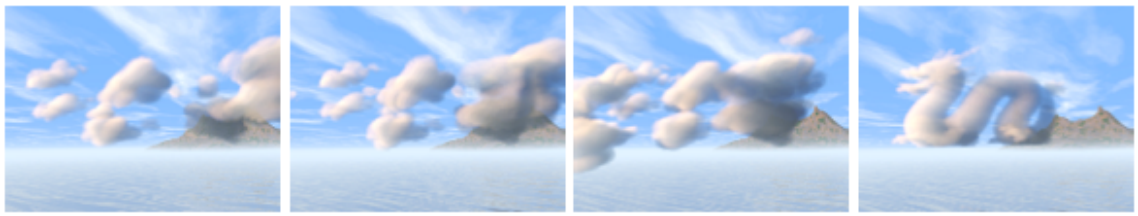
cloud is considered homogeneous. To model cumulus clouds, the shape is represented using an arbitrary triangle mesh where the interior of the mesh is set to possess a constant droplet density. At the boundary, density decreases when reaching the mesh surface, and increases when approaching the interior of the mesh. To achieve this, they stored the distance to the cloud boundary in the octree structure of voxels, as proposed by [CNLE09]. To further increase detail, fractal noise was added in the rendering step. Note that clouds generated in this manner are designed to be viewed from outside, yet they claim that their model can implicitly handle cases where the observer is inside the cloud. Taking into consideration that they used Gaussian textures, an artificial Gaussian blob is evident where some artificial non-cottony sharp effects are visible at the edge of the cloud shape.

More recently, Dreamworks Animation Studios produced a short film entitled [MMPZ12], in which clouds are generated using volumetric dynamic grids (VDB) [Mus13], as shown in Figure 2.12(g), which represent time-varying volumetric data discretized onto a 3D grid. Clouds are modelled as polygonal meshes and are scan-converted to level sets, represented by the VDB data structure. This level sets are then displaced using procedural noise to create a variety of textures from dense cumulonimbus to feathery nimbus

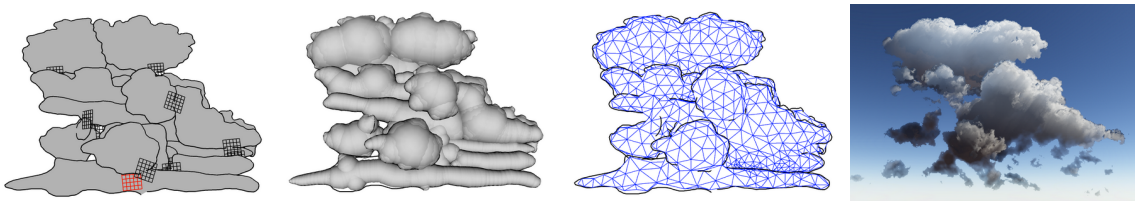
(a) [TBV02] (b) [BNL06] (c) [BNM+08]



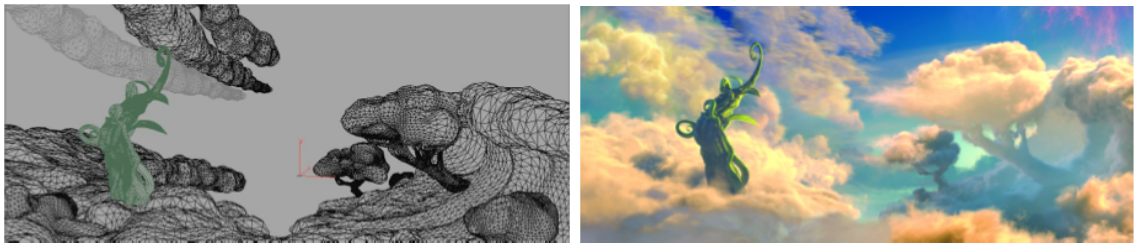
(d) Droplet trajectories from [SYS+06] method.



(e) Clouds generated using [LJW06] control method.



(f) [WBC08] method: the sketch of the cloud is defined, an implicit surface and a mesh are generated, and the final cloud is obtained.



(g) Clouds generated with [MMPZ12] method for the short film "Puss in Boots".

Figure 2.12: Examples of clouds generated with grid-based methods.

clouds.

To control the shape of clouds using implicit functions, Liu et al. [LJW06] used two geometric schemes to generate a cloud animation that matches a required shape in two different ways: by aggregation using several clouds and by diffusion from a single cloud. The first scheme simulates a large scale cloud animation, where several clouds are into the target shape progressively. To simulate aggregation, several blobs that approximate the target shape are subdivided into subsets. Next, the shape of the cloud that corresponds to each subset is modelled at several key-frames, being the cloud shape achieved by interpolating the positions and frames between keyframes. In the second scheme, cloud animation is produced in a forward manner, by progressively displaying the ellipsoid blobs. With this approach, they can simulate the diffusion effect of fluid dynamics [SY05].

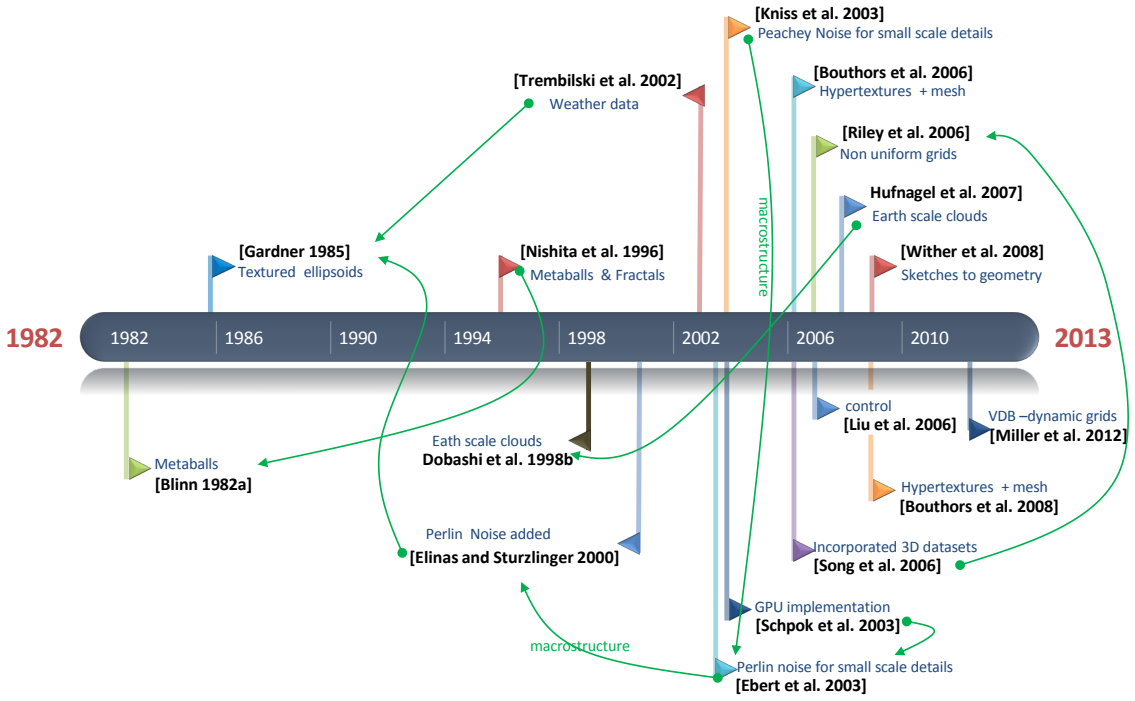


Figure 2.13: Timeline of volumetric methods used to simulate clouds. Green arrows represent relationships between methods.

Wither et al. [WBC08] took a slightly different approach in generation cloud shapes. Instead of generating directly the 3D shape of the cloud, their method provides the user with a tool to first create the 2D sketch of the cloud shape according to the approach presented by Alex [Ale05]. Over the sketch, a skeleton is inferred using a chord axis transform derived from a constrained Delaunay triangulation. This skeleton is segmented in terminal branches and body branches. This allows us to define volume features to be placed over the cloud shape. Once the outline of the cloud is defined, an implicit surface is generated and converted into a manifold mesh, as shown in Figure 2.12(f). Using this method, they are able to generate cloud meshes in about 2 minutes. The timeline of these methods is presented in Figure 2.13.

2.4 Image-based Methods

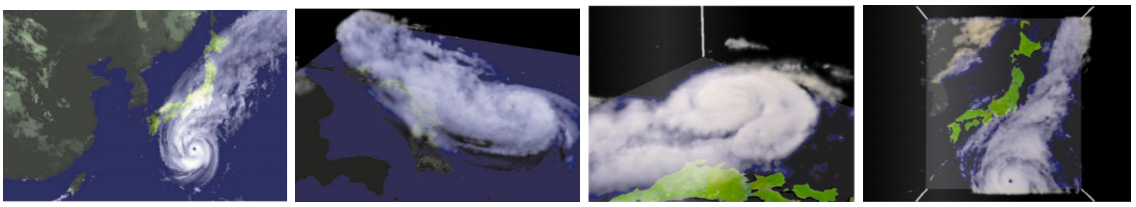
Cloud shapes can be generated using images or photographs. Interestingly, despite the extant geometry extraction algorithms from images in other fields other than computer graphics, only a few works of this sort have been carried in the field of cloud generation [DTM96, Qua10, PVGV+04]. In the literature, these works can be divided into two classes: satellite images and real photographs of clouds.

2.4.1 Satellite Images

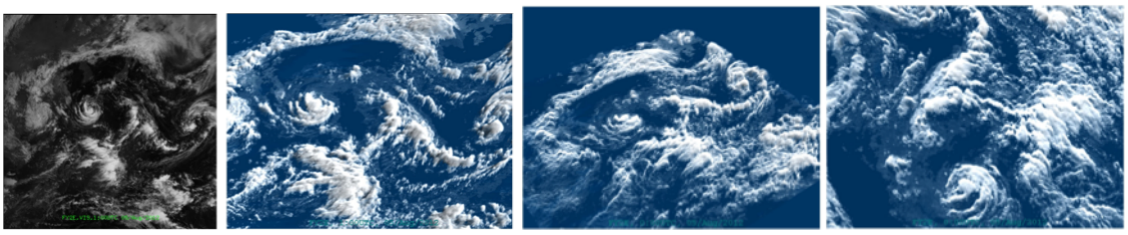
Lee et al. [LKS+96] were pioneers in modelling clouds from satellite images. In their method, these images are used as a basis to model earth scale cloud shapes using polygons. Nevertheless, their results are far from realistic, but a new trend in cloud simulation emerged with their work.

Dobashi et al. [DNYO98] modelled 3D clouds from satellite images by representing cloud densities as metaballs. Pixels in the satellite image are classified as being part of the cloud or not. After this, they generate a metaball at pixels that are part of the cloud and parameters of the metaball like radii, centre and density are automatically determined so that an image generated with metaballs matches the satellite image. Moreover, they simulate the evolution of a cloud system, by using sequences of satellite images. They applied their method to the visualisation of weather information, as shown in Figure 2.14(a).

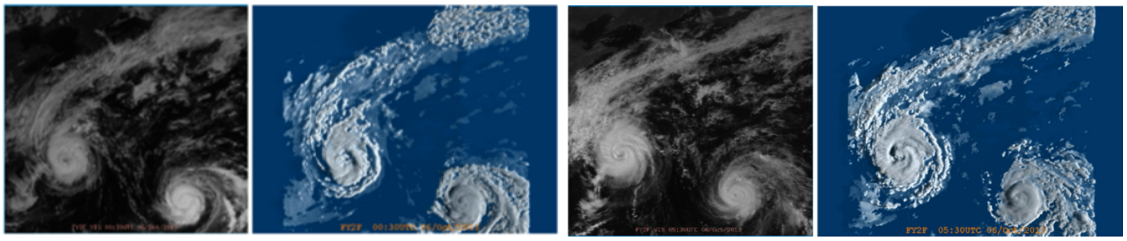
More recently, Yuan et al. [YLHY13] used five spectral images provided from a satellite to obtain cloud properties required for the generation of realistic earth scale clouds (Figure 2.14(b)). For each pixel, these images provide information regarding visibility, infrared, split, water vapour and infrared mid-wave. The first image (visibility) provides information about reflectance indicating the ratio of the reflected intensity to the incident sun light. The other four images record the temperature. These images are the basis of their method, which consists of four steps. First, pixels are classified into four types, namely ground, ice, thin cirrus and water, according to the information extracted from the infrared and split images. Next, the cloud top height is estimated using measured information obtained from visible and water vapour images. Finally, the cloud thickness and extinction are determined by empirical formulations. Liang and Yuan [LY15] improved on the cloud classification process and thickness estimation process for ice clouds and thin clouds, and a set of validation schemas are set up to evaluate cloud parameters such as cloud type, optical thickness and the effective radius. By using several images, they are able to simulate the evolution of the cloud system (Figure 2.14(c)).



(a) Satellite image used in [DNYO98] and resulting images in different views.



(b) satellite image used in [YLHY13] and resulting images in different views.



(c) Evolution of a typhoon using the method of [LY15] based on satellite images.

Figure 2.14: Clouds generated from satellite images. Credits to the images are of the authors of the referred papers.

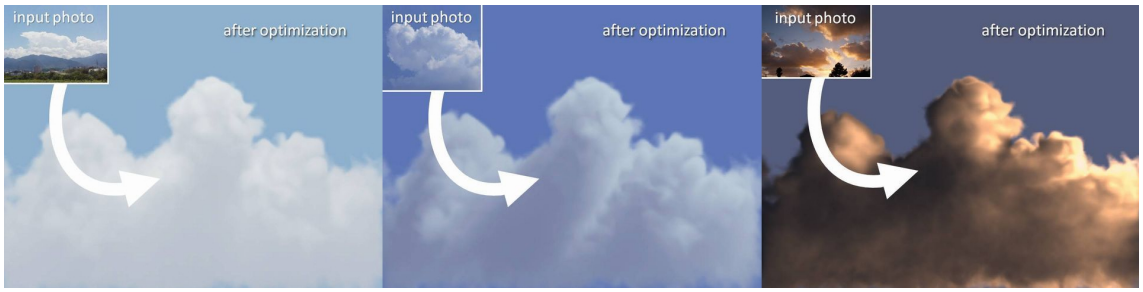
2.4.2 Photographs of Clouds

In this class, Dobashi et al. [DSY10] improved on a previous work [DNYO98] to account for the generation of several types of clouds. By determining densities, the computational cost of the previous method is reduced. In their method, three types of clouds are generated from photographs: cirrus, altocumulus and cumulus. They use a different approach for each type of cloud, with the common process of calculating the cloud image. From the input photograph, the sky image is extracted and the opacity and intensity of the image at each pixel is determined. This results in the cloud image. With this image, and depending on the type of cloud, one applies a distinct method. In the case of cirrus clouds, one uses a 2D texture from the cloud image. Altocumulus clouds are modelled using metaballs to define a density distribution with a field function [WT90]. As in a previous work due to Dobashi et al. [DNYO98], three parameters are determined to approximate the cloud image by these metaballs: centre position, radius and centre density. In the case of cumulus clouds (Figure 2.15(a)), the surface shape is generated by determining the thickness at each pixel (by determining the image cloud medial axis) and considering that clouds are thicker at the centre and thinner at the boundary, with Perlin noise being used to generate the density inside the cloud. In [DSY10, p. 2] the application of the method is described.

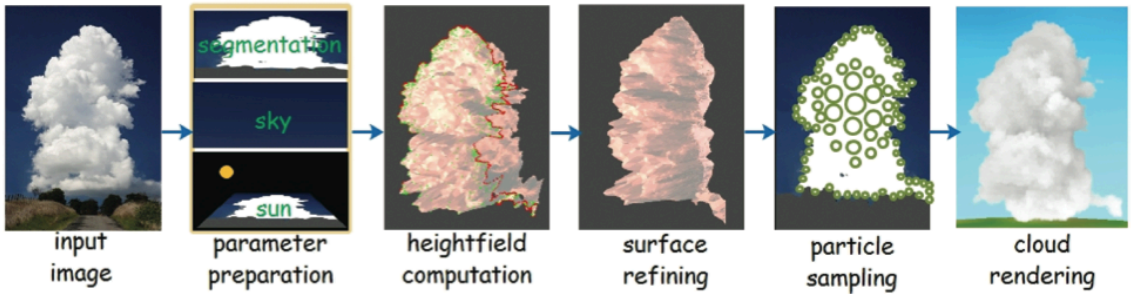
In a different approach, Dobashi et al. [DIO+12] mostly focused on cloud rendering from

photographs. In their method, a volume of data representing a given cloud, provided by the user, is used to match real photographs (Figure 2.15(b)). Their inverse rendering method is based on extracting the information from the photograph, using then this information to realistically render the volume of data.

(a) [DSY10]



(b) [DIO+12]



(c) [YLH+14]

Figure 2.15: Examples of clouds generated with images of real clouds.

Recently, Yuan et al. [YLH+14] improved on the cumulus clouds technique of Dobashi et al. [DSY10] to account for the recovery of details in the cloud surface. Their method is divided into five steps: parameter preparation, height field computation, surface refining, particle sampling and cloud rendering (cf. Figure 2.15(c)). For parameter preparation, first the image is divided into uniform subregions where the chroma-threshold method [DSY10] is used to form the exact boundaries of cloud images and sky images. The computation of the pixels height field is carried out by creating a cylinder that defines the local volume of a pixel. By determining the distance of the pixel to the sun, the volume changes according to that distance and the height field of a pixel is determined. After determining the height field, a symmetric cloud surface is obtained that requires refinement since flat regions are obtained in the surface from the height field computation. To refine the front surface, shape constraints are added, whereas the back

surface is obtained as a weighted sum of the height fields of the shape constraints. With the cloud surface calculated, the cloud is filled with particles generated through an adaptive sampling process. The rendering is carried out using the method proposed by Harris and Lastra [HL01]. Figure 2.16 shows a timeline of methods approached in this section, as well as their relationships.

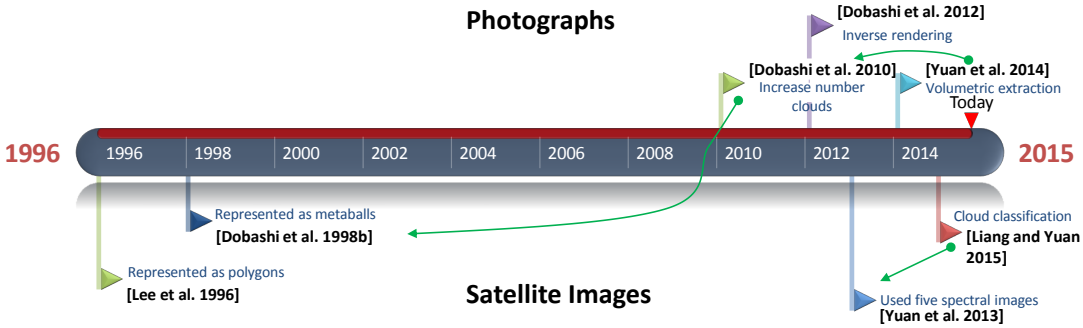


Figure 2.16: Timeline of methods based on images to simulate clouds. Green arrows represent relationships between methods.

2.5 Discussion and Future Trends

The variety of algorithms now available offers a number of solutions in terms of efficiency and visual quality, as shown in Figure 2.17 and Table 2.2. There are very high quality, though very slow, algorithms such as physically-based methods. But, there are also very fast algorithms as it is the case of those based on particle systems, but theses ones require tuning many parameters to simulate cloud dynamics. Interestingly, no effective tradeoff between these algorithms has ever found in the literature.

2.5.1 Discussion

Several key facts in cloud simulation have been discussed during the past decades. First, most literature is focused on cumulus clouds formation. This is a natural choice, since this type of clouds presents a pronounced volumetric appearance with flat bases and puffy domes which confer them a fluffy appearance. Second, it is very common to refer that “physically-based methods are computationally expensive” and “procedural methods require a trial and error approach” to simulate clouds. But, most works focus on procedural methods to simulate cloud formation. In fact, no intermediate approach has been developed to fully account for the benefits of both categories of methods. Third, most procedural methods fall into the category of volumetric/grid-based methods and particle systems. Interestingly, particle-based methods use volumetric methods (like volumetric implicits or grids) to represent the geometry of clouds. On the other hand, volumetric methods use particle systems to assist in the definition of the overall shape of clouds.

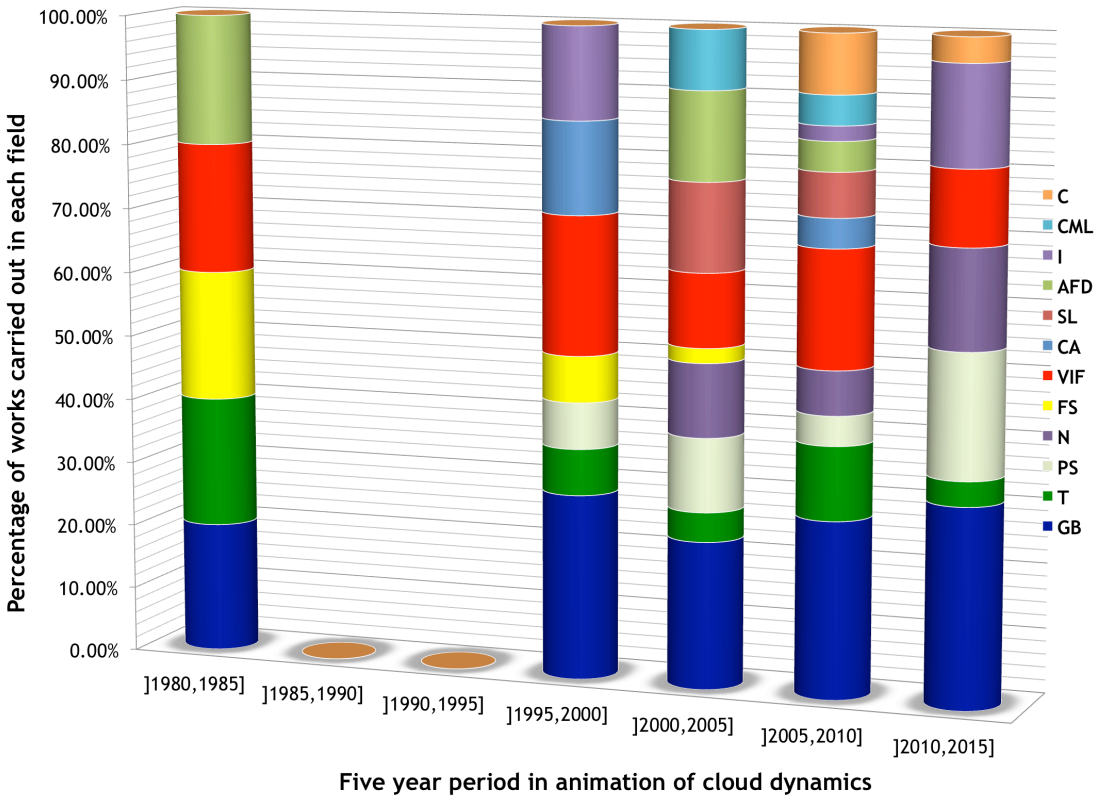


Figure 2.17: Graphical representation of methods developed per field of investigation, given the sample of works used in this survey.

Summing up, the most relevant features of different methods, as well as their advantages and disadvantages, are as follows:

- *Physically-based methods.* In the past eight years, with exception of the recent work of [BWDY15], no work has been presented in the field of computer graphics in physically-based cloud simulation, as shown in Fig 2.4. The computational cost associated with these methods led the computer graphics community to find faster solutions, namely using particle systems and volumetric methods. However, this is still an open issue, since methods that use the CML method are still used in commercial packages, such as Siverlinning, by Sundog Software [Sof15]. Therefore, reducing the complexity of physically-based methods is an important issue in computer graphics.

  Another issue is the ability to control the fluid shape during simulation. Methods like those proposed in [MDN02] and [DKNY08] focused on this, but the first uses techniques that make it difficult the task of artists to define the shape of clouds, while the second controls the formation of clouds from a 2D silhouette. Despite the potentiality of this method to control the overall shape of clouds, two major questions remain unanswered: can it be adjusted to three-dimensional shapes? Can these clouds have horizontal motion? Although the parameters in the Navier-Stokes equations such as viscosity and body forces give the animators

some degrees of shape control, higher-level control is also desired in a production environment, where animators are mostly interested in modifying the large-scale motion of the fluid, while the physically-based simulation should take care of the fine-scale details such as small vortices. Therefore, there is a lack of techniques that allow a better control of cloud formation (and consequently define shapes of clouds) in physically-based simulation methods.

- *Implicit geometries.* A brief glance at Figure 2.17 shows that that most methods to simulate clouds fall into the category of geometric-based methods, which are mostly based on volumetric implicit functions (VIF) or grids (GB). Due to the complexity of cloud motions, it is non-trivial to model the cloud surface by explicit triangle meshes since cloud is likely to merge or split, which will invalidate the topologies of original mesh during the simulation. Nevertheless, this technique is often complemented with noise to account for details at the microstructure level [SSEH03, EMP+03, KPH+03, BNM+08].

- *Particle Systems.* In cloud simulation, particle systems would be a natural method to simulate cloud formation, since clouds are composed of particles of dry or moist air. However, particle systems were mostly used to define the overall shape of clouds, instead of its dynamics. Interestingly, methods based on particles systems often require a mesh for post-processing features and methods based on implicit geometries use particles to obtain desired shapes of clouds. But, since most rendering methods are grid-based [HH12], there is the need to make such transformations, which incur in efficiency costs. Another classic drawback of particle-based methods is scalability. More particles imply more detail over the cloud shape, but also implies on a huge increase on the computational cost associated to simulate the formation of clouds. To overcome this constraint, the typical solution is to generate a mesh and apply noise and wispy to it in order to endow more detail to the cloud boundaries.

- *Image based.* Most image based methods focus on extracting information from images and converting this into a mesh. Typically, they use one image or a set of images with different types of information to infer volumetry and density of clouds. However, with two dimensional images, a three-dimensional shape of a cloud is difficult to achieve. Moreover, it is very difficult to infer motion of clouds from images. It requires several sequential images from which information is extracted, and then small simulations can be obtained. However it is computationally expensive, with the aggravating circumstances that the resulting clouds suffer from lack on temporal coherence.

The aforementioned features represent the macroscale advantages and disadvantages of using the surveyed methods. Most of the features and limitations provided are also hardware associated. Therefore, one has also to consider that technology is continuously evolving, so that it is expected that GPU implementations (and the like) will play

Table 2.2: Representation of methods used in cloud simulation. Classification is established regarding the type of simulation, the type of clouds and specific techniques used. Techniques related to simulation are marked with ● and those that are more related to rendering are marked as ●. The abbreviations represent: **GB**: Grid-based; **T**: Textures; **PS**: Particle systems; **N**: Noise; **FS**: Fourier synthesis; **VIF**: Volumetric implicit functions; **CA**: Cellular automata; **SL**: Semi-lagrangian; **AFD**: Atmospheric fluid dynamics; **I**: Images; **CML**: Coupled map lattice; **F**: Fractals; **WM**: Wind model; **GPU**: GPU implementation; **C**: Control; **STP**: Similarity turbulence plume; **S**: Sketches; **VDB**: Volumetric dynamic grid; **PBF**: Position based fluids.

Year	Reference	Animation Type		Types of Clouds					Methods											
		Procedural	Physics	Cumulus	Stratus	Cirrus	Volcanic	Earth	GB	T	PS	N	FS	VIF	CA	SL	AFD	I	CML	Other
1984	Kajiya and Von Herzen [KVH84]		⋆	✓					●								●			
1985	Gardner et al. [Gar85]	⋆		✓	✓	✓				●			●	●						
1996	Lee et al. [LKS+96]	⋆						✓	●									●		
1996	Nishita et al. [NDN96]	⋆		✓					●					●						F
1998	Dobashi et al. [DNO98]	⋆		✓					●						●					
1998	Dobashi et al. [DNYO98]	⋆						✓						●				●		WM
2000	Dobashi et al. [DKY+00]	⋆		✓					●	●					●					WM
2000	Elinas and Stürzlinger [ES00]	⋆		✓	✓	✓				●			●	●						GPU
2001	Miyazaki et al. [MYND01]		⋆	✓					●							●	●		●	WM
2001	Harris and Lastra [HL01]	⋆		✓							●									
2002	Miyazaki et al. [MDN02]		⋆	✓					●							●	●		●	WM
2002	Overby et al. [OMK02]		⋆	✓	✓				●							●	●			
2002	Trembilski et al. [TBV02]	⋆		✓	✓	✓							●	●						WM
2003	Harris et al. [HBSL03]		⋆	✓					●							●	●			GPU/WM
2003	Wang [Wan03]	⋆		✓	✓	✓				●	●	●								WM
2003	Riley et al. [REHL03]	⋆		✓							●	●								
2003	Ebert et al. [EMP+03]	⋆		✓	✓	✓						●		●						WM
2003	Kniss et al. [KPH+03]	⋆		✓	✓	✓				●		●		●						
2003	Schpok et al. [SSEH03]	⋆		✓	✓	✓						●		●						GPU/WM
2004	Mizuno et al. [MDN04]		⋆				✓		●							●	●		●	WM
2004	Geist et al. [GRWS04]		⋆	✓					●							●	●		●	
2004	Bouthors and Neyret [BN04]	⋆		✓							●			●						
2005	Umenhoffer and Szirmay-Kalos [USK05]	⋆		✓						●	●									
2006	Dobashi et al. [DYN06]		⋆					✓	●							●	●		●	C
2006	Liu et al. [LJW06]	⋆											●	●						C
2006	Riley et al. [RSK+06]	⋆		✓	✓	✓		✓	●				●							GPU
2006	Song et al. [SYS+06]	⋆		✓	✓	✓			●				●							GPU
2006	Bouthors et al. [BNL06]	⋆			✓				●				●							GPU/WM
2007	Wang et al. [WPKK07]		⋆	✓					●							●	●			C/STP
2007	Grudzinski and Debowski [GD07]	⋆		✓							●									
2007	Hufnagel et al. [HHS07]	⋆						✓						●						
2008	Dobashi et al. [DKNY08]		⋆	✓					●							●	●		●	C
2008	Shen et al. [SYL08]	⋆		✓										●	●					WM
2008	Wither et al. [WBC08]	⋆		✓										●						S
2008	Bouthors et al. [BNM+08]	⋆		✓										●						GPU
2009	Cha et al. [CSI09]	⋆		✓							●			●						GPU
2010	Upchurch and Semwal [US10]	⋆			✓	✓									●					WM
2010	Dobashi et al. [DSY10]	⋆		✓	✓	✓			●					●						
2011	Yu et al. [YW+11]	⋆		✓	✓	✓														
2012	Wenke et al. [WSYM12]	⋆						✓												
2012	Do et al. [DBLR12]	⋆		✓	✓	✓														
2012	Dobashi et al. [DIO+12]	⋆		✓	✓	✓	✓	✓	●											
2012	Miller et al. [MMPZ12]	⋆		✓	✓	✓								●						VDB
2013	Yuan et al. [YLHY13]	⋆						✓	●											
2014	Yusov [Yus14]	⋆		✓																GPU
2014	Yuan et al. [YLH+14]	⋆		✓					●					●						
2015	Barbosa et al. [BWDY15]		⋆	✓																C/GPU/PBF
2015	Liang and Yuan [LY15]	⋆						✓	●											

a key role in the production of even more realistic and complex simulations of cloud systems.

2.5.2 Future trends

Considering the set of features and limitations presented earlier, several techniques have been identified to overcome such limitations:

- *Real-time simulation.* Real-time applications such as video games require the integration of clouds with their dynamics, in order to have realistic rendering of cloudy scenes. In this respect, GPU computing based solutions seem to be the vehicle to achieve such realism in real-time, as well as to diminish the visual gap between real scenes and synthetic scenes.

- *Real data to simulate cloud formation.* Several methods used real data provided by weather agencies, namely for particle systems [REHL03, WSYM12] and volumetric methods [DNYO98, TBV02, RSK+06, SYS+06]. However, these works use large datasets made available by weather agencies which include a huge amount of information. This makes these methods difficult to generate clouds in real-time. Other types of data should be exploited that focus not only on direct cloud representation from a dataset, but provide the ability to simulate cloud formation. Many weather agencies provide data from radiosonde balloons which establish temperature profiles of the atmosphere (e.g., http://www.twisterdata.com and http://www.woweather.com). From these, we may be able to determine the temperature profile of the air mass that composes a cloud. This will make it possible to simulate cloud formation. Since these data are available at every 6 hours, it may also account for cloud dynamics. Recently, the concepts associated to temperature profiles of the atmosphere were used to characterise the evolution of weather in an environmental model of snow accumulation [MGG+10].

- *Large scale simulation.* In movies and games, massive requirements of large scale simulation of clouds are required. Many times they are treated as sky textures, which does not make possible real interactions with them. When clouds are treated as grids or meshes, a simple scaling operation on the cloud field maintaining the same grid resolution will be problematic since computational costs will increase dramatically. So far, methods that account for this scaling require parameter pre-computation [CSI09]. One solution is to partition the space domain into subspaces and simulate each subdomain with an adequate solver. For example, far clouds can be simulated with textures or impostors (as already referred by [Yus14] as future work), while near clouds can be simulated with volumetric implicits or particle systems.

- *Multiresolution.* Large scenes composed of several types of clouds, which are represented using meshes, typically have large amounts of geometry detail. This

compromises efficiency and interactivity, not to mention the scalability issue. One general concept which is widely accepted as fundamental to the construction of scalable algorithms is the idea of multiresolution [DFS05], namely in mesh simplification. These techniques have already been successfully applied to terrain modelling [PG07], and fluids [KAD+06]. Cloud simulation could also gain from this field by simulating clouds that account for different viewpoints: close and far, at different resolutions, depending on the user position.

2.6 Concluding Remarks

In this chapter, we have presented an overview of methods focused on cloud simulation, as well as pointed out several directions for future work, taking into consideration the advantages and disadvantages of different approaches in cloud simulation.

Initially, a background on clouds was presented to provide the reader with some insight of the different types of clouds and their formation process. Also, we put forward a taxonomy of cloud simulation methods in computer graphics. Such taxonomy divides such methods into three major categories: physically-based methods, procedural methods, and image-based methods. Then, we carried out a comprehensive analysis of each category of methods, including a timeline that shows the relationships between such methods over time. In addition, Table 2.2 further details such simulation methods, as well as the types of clouds they are able to produce.

This classification made it possible the identification of features inherent to simulation methods, with their advantages and disadvantages. Most methods fall into the category of procedural volumetric methods, which aim at simulating cumulus clouds, because they are those that are immediately observed by a character standing or walking on the ground.

In the near future, it is expected that cloud simulation can improve on scalability by introducing multiresolution techniques capable of handling large cloud scenes and to incorporate real data to simulate cloud evolution. This last trend may fill the gap between physically-based methods and procedural methods, by “feeding” discrete forms of the Navier-Stokes equations with sounding data to achieve real-time simulation times.

In short, on the issue of “How is it possible to simulate cloud formation in computer graphics?”, we can say that current approaches offer a wide variety of methods that can be combined to effectively simulate the full cycle of cloud formation: birth, evolution and death. But, a number of issues have to be overcome to attain higher degrees of visual realism at interactive rates.

Submitted Paper

The work described in this chapter originated a survey on methods used in cloud simulation, which was submitted for publication as follows:

> **Rui P. Duarte, Francisco Morgado and Abel J.P. Gomes: Techniques for Cloud Simulation in Computer Graphics: a Survey, *ACM Computing Surveys* (submitted for publication).**

Chapter 3

Real-Time Simulation of Cumulus Clouds through SkewT/LogP Diagrams

The modelling, simulation, and realistic rendering of natural phenomena have been important goals in computer graphics for decades, largely because of their multiple applications in film animation, virtual environments and video games. Clouds, as a natural phenomenon, represent a real challenge because their formation (birth), movement (life), and extinction (death) are amorphous and dynamic in nature. As a consequence, cloud simulation is computationally very expensive when using physically-based methods. In order to overcome this problem, this chapter presents a computationally inexpensive, real-time method for the simulation of clouds using SkewT/LogP diagrams. These diagrams provide us with a 2D procedural technique to simulate 3D clouds from sounding data made public worldwide by weather agencies. To achieve real-time rates, our system is based on physics, but avoids solving differential equations. We have also built a visual tool for 2D SkewT/LogP diagrams that allows us to inspect, control and simulate the thermodynamic process of ascending clouds in the atmosphere, as well as a 3D synthetic environment where clouds are advected by buoyant forces. This lightweight procedural technique enables the incorporation of our cloud simulator in systems tied to a number of industries, namely computer games and movies.

3.1 Introduction

The realistic simulation of clouds in synthetic environments has always been a hot research topic in computer graphics. However, simulation of clouds on computer is a complicated task since they have physics-dependent dynamic shapes that evolve over time. Cloud simulation involves two distinct fields: computer graphics and meteorology. While the first focuses on representing clouds in a visually appealing way, the second focuses on the physics of clouds and weather forecasting. Both perspectives are integrated in our cloud simulator. It takes advantage of thermodynamic diagrams, also called SkewT/LogP (Skewed Temperature and Logarithmic Pressure) diagrams [AWSH61], to quickly solve partial differential equations that model the flow of clouds as fluids and generates cloud shapes and dynamics that can be easily integrated into flight simulators and computer games. As illustrated in Figure 3.1, the main contributions of the work carried out in this chapter are the following:

- *SkewT/LogP diagrams.* To our best knowledge, this is the first cloud simulator that uses SkewT/LogP diagrams in computer graphics. One of the key prob-

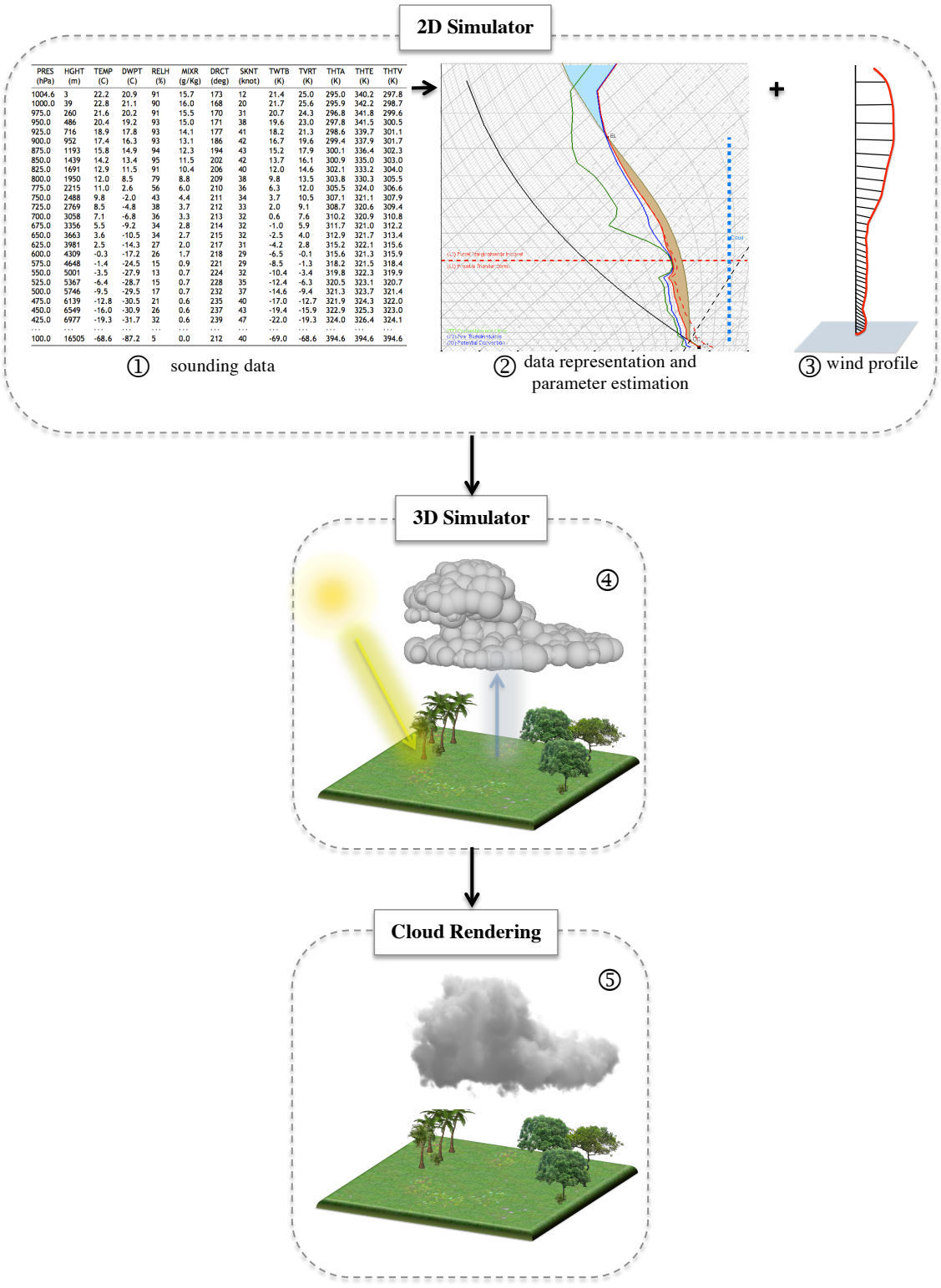


Figure 3.1: Overview of the SkewT/LogP system: ① input atmospheric sounding data collected by weather forecasting agencies around the planet; ② sounding data feed the SkewT/LogP diagram in order to estimate atmospheric parameters associated to a set of clouds; ③ sounding data also allows us to generate the wind profile associated to a set of clouds; ④ particles of each air parcel that originates a cloud rise in the atmosphere; ⑤ each air parcel starts the condensation process at some point in the atmosphere, so that the corresponding cloud becomes visible and needs to be rendered on screen.

lems in cloud simulation is to establish a realistic profile for the environmental temperature, cloud base, and cloud top. Instead of using fixed lapse rates for the environmental temperature [Ney97, GD07], we use sounding data obtained from real weather forecasts, namely those taken from http://weather.uwyo.edu, http://www.twisterdata.com, and http://www.woweather.com. These data include diverse temperature profiles, which allow us to automatically determine the cloud base and cloud top through geometric intersections in the SkewT/LogP diagram. This approach avoids a common problem in procedural methods that is the trial and error process of tuning atmospheric parameters.

- *Explicit solution of the equation of motion for clouds in real-time*. The cloud formation process is mostly related to thermal convection and atmospheric stability, and closely related to an important atmospheric property known as vertical motion. By using SkewT/LogP diagrams, we were able to explicitly determine the vertical rising force required to solve the equation of motion that characterises the vertical displacement of an air parcel (mass of air) in the atmosphere, without solving the corresponding differential equations.

- *Automatic generation of clouds that adapt to the weather conditions*. Our method improves on methods that use manual modelling of clouds shape. It can be considered a plugin replacement either for artists that focus on cloud rendering, or for the simulation of cloud systems for flight simulators or games. By using real data, we are able to generate clouds that adapt to the weather conditions.

The remainder of this chapter is organised as follows. Section 3.2 briefly reviews related works in cloud simulation. Section 3.3 reviews the physics of clouds, in particular the equation of cloud motion. Section 3.5 details the curves in the SkewT/LogP diagram needed to solve the equation of cloud motion. Section 3.6 describes the formation of clouds in three stages: birth, dry lift, and moist drift. Section 3.7 presents relevant results produced by our cloud simulator. Section 3.8 approaches the limitations of the method in the simulation and modelling of clouds over time. Section 3.9 concludes the chapter, providing some hints to future work.

3.2 A Brief Overview of Related Works

The dynamics and appearance of clouds have received a lot of attention from the computer graphics community in the last three decades. In this section, we briefly review the most relevant techniques in cloud simulation. For a more detailed description of the related work, the reader is referred to Chapter 2.

Cloud simulation is usually based on either the physics theory of fluid dynamics or on procedural techniques. Most physically-based methods involve solving partial differential equations (PDEs), in particular the Navier-Stokes equation, which is very expen-

sive in computational terms. Simplified physically-based methods, such as stable fluids [Sta99] and coupled map lattice (CML) [MYND01, HBSL03, GSW07], have been proposed in computer graphics, yet they have trouble achieving real-time rates in rendering.

Procedural methods in cloud simulation represent an attempt to avoid the computational burden inherent to physically-based methods. Typically, these methods generate the cloud density distribution, using the idea of fractals [Vos85], noise [Per85, SSEH03, Man06], Fourier synthesis [Gar85, ES00], volumetric implicit functions [EMP+03, WBC08, TBV02], cellular automata [DKY+00], and particle systems [Ney97, BN04, GD07]. Interestingly, Schpok et al. [SSEH03] used a two-level procedural approach for modelling and animating several types of clouds. At the first level, they used volumetric procedures together with noise and turbulence data to produce various types of clouds by applying different noise filters [Per85, Gre05]; at the second level they use volumetric implicits [Gar85, ES00] to model the clouds at the second level. More recently, Dreamworks Animation Studios produced a short film [MMPZ12], where clouds are generated using volumetric dynamic grids [Mus13] which represent time-varying volumetric data discretized on a 3D grid.

In the area of cloud simulation from weather data, little work has been carried out. Trembilski et al. [TBV02] generated surface clouds from data produced by routine meteorological weather simulation. The cloud isosurface is evaluated, sampled and displayed using the marching cubes algorithm. Wenke et al. [WSYM12] produced world view clouds where each grid point is treated as a particle. Their method focuses on modelling cloud shapes, not on cloud evolution in the atmosphere.

Our approach uses particle systems, and considers thermal advection based on the vertical buoyancy forces and horizontal wind forces, obtained from weather data. In this respect, our approach is quite different from others found in the literature. Neyret [Ney97] generates clouds considering that the altitude at which a cloud forms is constant and, thus, known beforehand, also assuming that the environmental lapse rate and the air parcel lapse rate are constant. In our method, these parameters are automatically determined from the sounding data for each particle in the ascending air mass. Grudzinsky [GD07] uses an exponential function to generate trajectories for the air parcels, and a cloud generation function to assign a set of parameters to each particle. On the contrary, our approach automatically determines particle trajectories from thermodynamic diagrams, also known as SkewT/LogP diagrams.

3.3 Cloud Physics

When one looks at the sky and observes the different shapes, sizes and appearance of clouds, it seems almost impossible to categorise them. This classification is determined and updated by the World Meteorological Organization (WMO), and serves as an international guideline for the classification of clouds observed all over the world [Mas57]. In

this chapter, we focus on *cumulus clouds*, which usually result from the upwards motion (also called convection) of air parcels.

3.3.1 Parcel Theory

Conceptually, parcel theory describes the upwards motion and changes of a parcel of air in the atmosphere (see Figure 3.2). The lifting of an air parcel is done by buoyant and mechanical forces, which cause the expansion of the air parcel as a response to a decrease in pressure and temperature. Eventually, this air mass becomes saturated (i.e., its relative humidity becomes 100%) so that it condenses into droplets to form a cloud. Saturation can occur by the way of atmospheric mechanisms that cool off an air mass to its dew point or frost point. This releases latent heat that warms the air parcel, increasing the parcel buoyancy and causing it to rise higher in the atmosphere. When the air parcel reaches its maximum altitude, it starts descending as a consequence of negative buoyancy until equilibrium is reached.

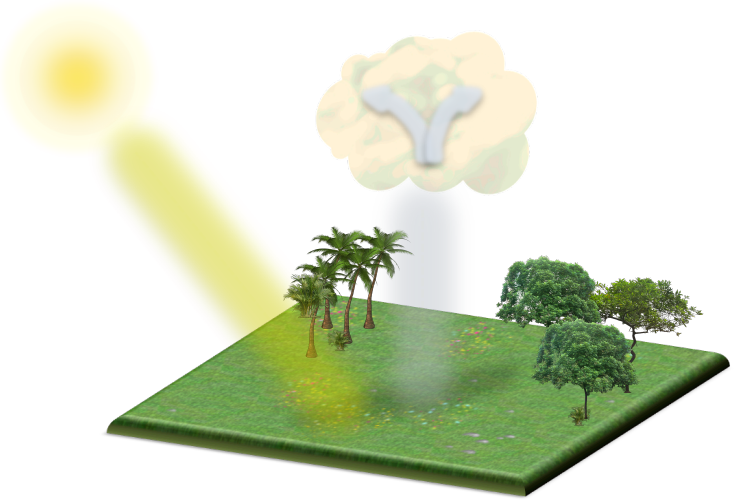


Figure 3.2: Parcel theory: the sun warms Earth surface and, consequently, a parcel of warm air rises into the atmosphere. When this parcel rises, it expands and cools. Water vapor condenses out from the cool air giving rise to a cloud. Once such a cloud reaches its maximum altitude, the cloud spreads out or begins descending.

3.3.2 Equations of Cloud Motion

The convection process seems quite intuitive, but it is governed by the laws of thermodynamics. Understanding these laws helps in predicting the formation of clouds, as well as in making clear how the vertical profile of atmospheric temperature is determined.

Before condensation, air has no liquid water, i.e., we have dry air. A dry air parcel is described by three properties: temperature (T, in K), pressure (P, in N/m^2) and density

(ρ, in kg/m^3). These properties are related by the ideal gas law [CBK11]:

$$P = \rho R T \tag{3.1}$$

where R is the gas constant that takes the value $R_d = 287.05307\,\text{Jkg}^{-1}\text{K}^{-1}$ for dry air, and $R_m = 461.5\,\text{Jkg}^{-1}\text{K}^{-1}$ for moist air. As the air parcel expands in response to a decrease in pressure, its temperature also decreases, resulting in a process called *adiabatic expansion*, which is governed by the first law of thermodynamics [CBK11].

In the vertical direction, gravity is the most important external force acting on the atmosphere. The balance between pressure and gravity is known as the hydrostatic balance. The environment pressure P_e and parcel pressure are said to be in hydrostatic balance when

$$\left(\frac{dP_e}{dz}\right)_z \approx -\rho_e(z)g \approx -\rho_p(z)g, \tag{3.2}$$

where g is the acceleration due to gravity ($g = 9.8m/s^2$), ρ_e and ρ_p stand for the environment and parcel densities, respectively. The upward force is just the environmental pressure gradient, $(dP_e/dz)_z$ and the downward force is $-\rho_p(z)g$. From Newton's law of motion [Cro00], we have

$$\rho_p(z)\frac{dv}{dt} = -\left(\frac{dP_e}{dz}\right)_z - \rho_p(z)g \tag{3.3}$$

where dv/dt denotes the vertical acceleration of the air parcel.

Equation (3.3) is the basic equation of motion for the air parcel at level z, which expresses the vertical acceleration of the air parcel as the vertical gradient of the pressure perturbation and buoyancy. Once the parcel has been lifted to z, it has the same pressure as the environment, that is, $P_e(z) = P_p(z) = a$. Using the ideal gas law (Equation (3.1)), we have $\rho_e(z) = (a/R)[1/T_e(z)]$ and $\rho_p(z) = (a/R)[1/T_p(z)]$. Considering this and substituting (3.2) in (3.3), we obtain:

$$\frac{dv}{dt} = g\Big[T_p(z) - T_e(z)\Big]/T_e(z) \tag{3.4}$$

The ideal gas law and the laws of thermodynamics can be used to derive Poisson's equation for adiabatic processes, which relates the temperature and pressure of a gas under adiabatic changes [RY89]:

$$\left(\frac{T}{T_0}\right) = \left(\frac{P}{P_0}\right)^k, \tag{3.5}$$

where T_0 and P_0 are the initial values of temperature and pressure, and T and P are the values after an adiabatic change in altitude, with

$$k = \frac{R_d}{c_p} = \frac{c_p - c_v}{c_p} \approx 0.286, \tag{3.6}$$

where c_p and c_v are the specific heat capacities of dry air at constant pressure and volume, respectively.

In atmospheric sciences, potential temperature is a more convenient variable to account for adiabatic changes of temperature and pressure, since its value is constant under adiabatic change of altitude. The potential temperature θ of a parcel can be defined as the temperature that the air parcel would have if it were moved adiabatically from pressure P and temperature T to pressure P_0. It is given by the following expression:

$$\theta = T\left(\frac{P_0}{P}\right)^{R_d/c_p} \tag{3.7}$$

By using potential temperature instead of absolute temperature in Equation (3.4), air parcels of different altitudes can be compared regardless of the corresponding air pressure, so we have the final form of the equation of motion in terms of potential temperature:

$$\frac{dv}{dt} = g\Big[\theta_p(z) - \theta_a(z)\Big]/\theta_a(z) \tag{3.8}$$

Therefore, Equation (3.8) can be used to determine the vertical displacement of an air parcel in the atmosphere by determining the potential temperature of both air parcel and ambient temperature, at a given pressure level. However, to speed up our computations, we use the SkewT/LogP diagrams to calculate those two potential temperatures instead, as explained further ahead.

3.4 Wind

Wind is considered a horizontal force in the atmosphere since its initiating factor is due to horizontal temperature differences in the troposphere. Air masses will have different densities and this causes pressure differences which has a great impact on the small scale motion of clouds. Interestingly, sounding data provide information that can be used to obtain a realistic vertical distribution of wind in the atmosphere, namely columns 7 and 8 of the atmospheric soundings presented in Table 3.1. These columns relate to parameters that allow us to determine the horizontal displacement related to wind: the wind direction α (7th column, in degrees, clockwise from true north) and wind intensity or speed r (8th column, in knots).

3.4.1 Hodographs

Before establishing the relation of sounding data with wind profiles, let us refer a tool commonly used by meteorologists to represent and study wind in the atmosphere: hodographs (see Figure 3.3). Invented by Hamilton in 1847 [Ham47] to solve the problem of deducing the law of gravitation that makes planets revolve in elliptical orbits

around the sun, it has been used in meteorology to forecast weather based on wind profiles of the atmosphere.

The structure of an hodograph is composed of two scales: angle and intensity. Around the hodograph, the grid is split in angles that go from 0° to 360°. Note that a direction of 180° is at the top of the diagram. This axis rotation is used to make it easier to plot a hodograph from upper air data. Recall the 180° wind comes from the south but its vector representation points to the north. North is usually placed at the top of the page, as shown in Figure 3.3. The intensity scale gives the length of the wind vector. It is represented in the (x, y) axes of the hodograph.

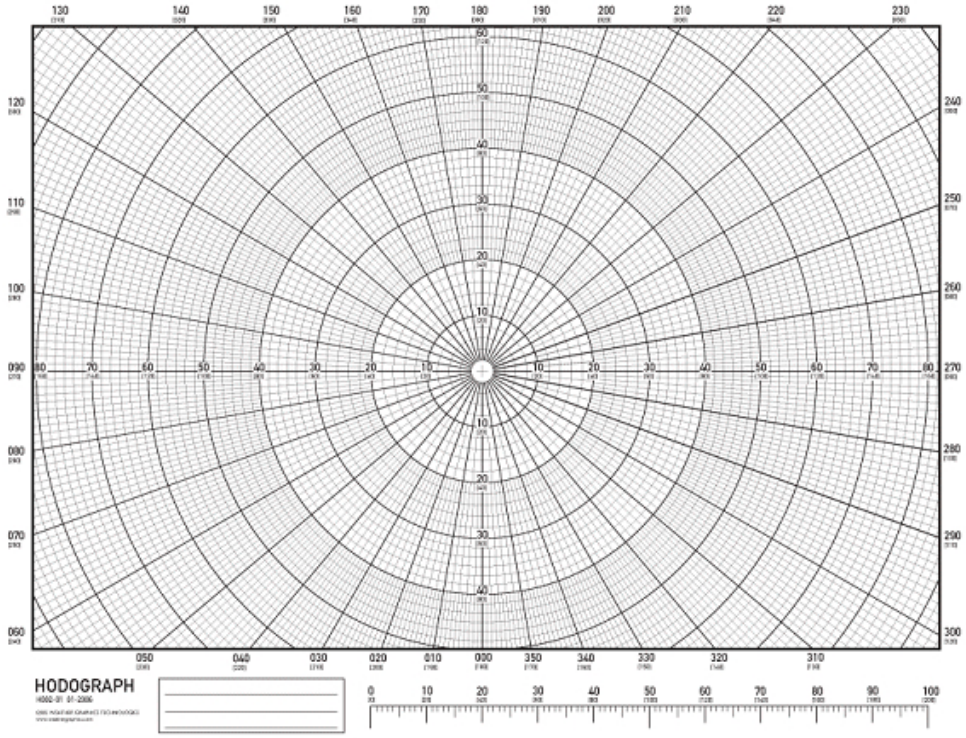


Figure 3.3: A blank Hodograph.

By reading a hodograph, meteorologists are able to determine the potentiality of thunderstorm formation, since it displays the change of wind speed and direction with height (vertical wind shear) in a simple diagram. Wind speed and directions are plotted as vectors with tails at the origin and the point in the direction toward which the wind is blowing. The length of arrows is proportional to wind speed. With simple geometric representations, they provide answers to problems that otherwise would required more complex formulations.

3.4.2 Plotting Wind Data

The wind vector represented in a hodograph is given by the following equations:

$$\begin{aligned} \Delta x &= r\cos(\alpha) \\ \Delta y &= r\sin(\alpha), \end{aligned} \tag{3.9}$$

and the process is illustrated in Figure 3.4. Consider, for example, an angle $\alpha \in [0°, 90°]$, wind blowing at intensity r. First, the angle is plotted in the graph, then the line segment that connects the origin to the intensity related to α determines the wind vector at height z.

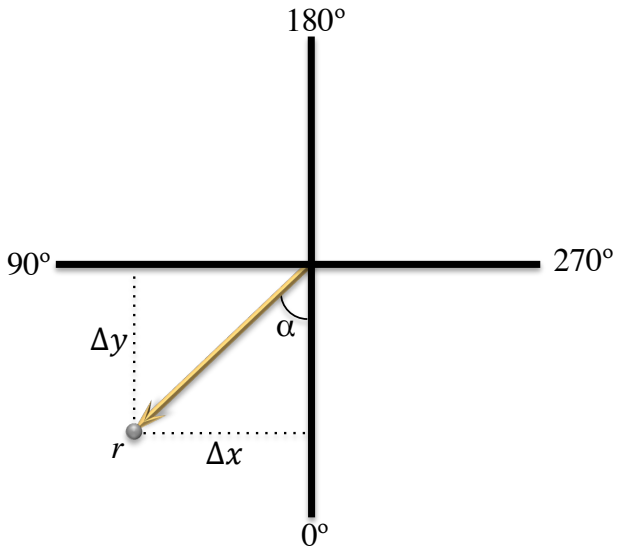


Figure 3.4: Representing wind sounding data in an hodograph for an arbitrary direction $\alpha \in [0°, 90°]$.

The process is repeated for all heights in a sounding and the final representation of the wind profile is obtained in the hodograph, as shown in Figure 3.5.

Reading a hodograph allows meteorologists to infer the type of wind behaviour in the atmosphere. From the example illustrated in Figure 3.5, the wind profile turns clockwise or veers with height, which is typically associated to warm air advection. It is called a veering wind.

3.4.3 3D Vertical Wind Profile

As seen in Figure 3.5, we are able to represent the wind data into a 2D diagram that also represents the vertical wind shear in the atmosphere. However, we require to have a 3D vertical representation of the wind profile to assist in the evolution of clouds in the atmosphere, so that particles at a given position can be moved by the corresponding wind vector. Therefore, we have extended the 2D model to a 3D profile of wind, by creating 3D planes of hodographs at each altitude from the sounding. The resulting wind shear curve allows us to interpolate particle pressures and determine the corresponding wind

height (km)	DRCT (degree)	SKNT (knots)
0	121	20
1	175	35
2	229	52
3	258	55

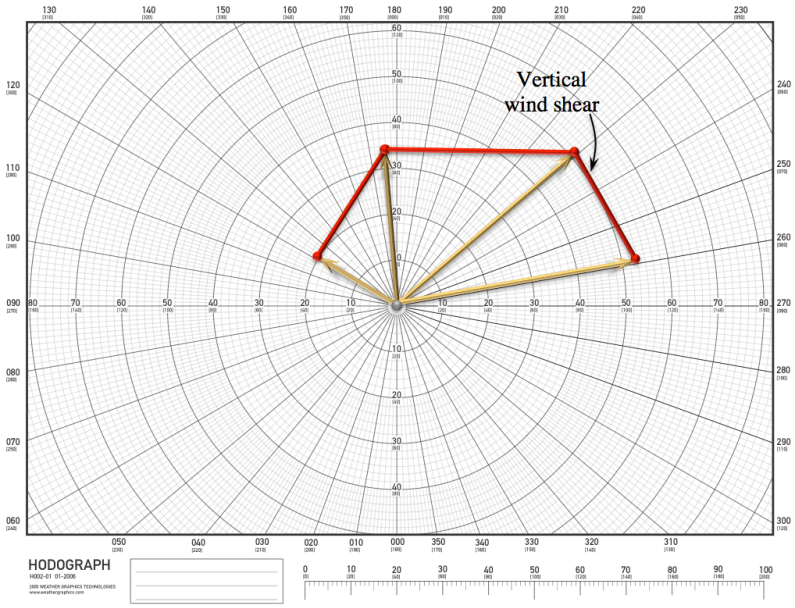


Figure 3.5: Wind sounding data and the corresponding representation in an hodograph using Equation 3.9.

vector. With this, we are able to determine the particle's horizontal displacement during advection in the atmosphere, at each pressure level (and, consequently, altitude), as shown in Figure 3.6.

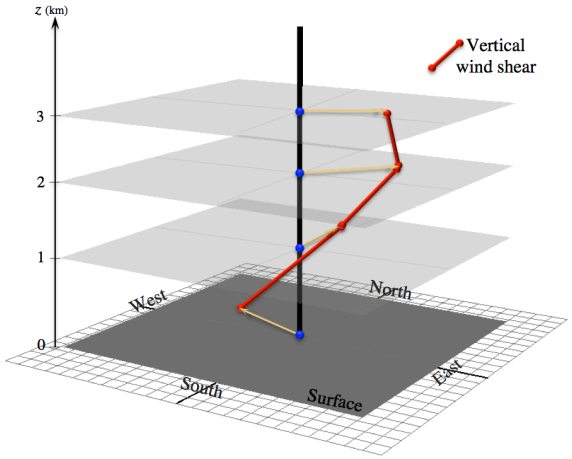
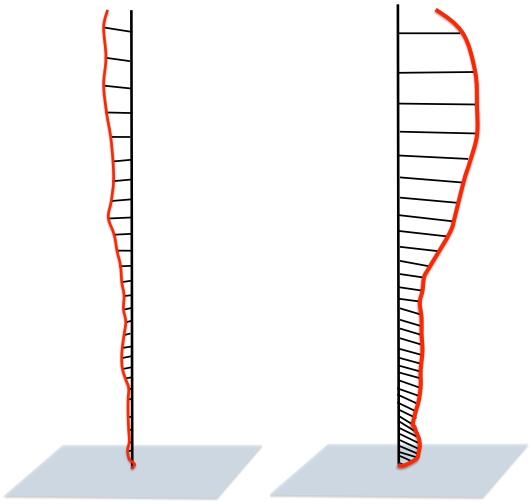


Figure 3.6: Vertical wind shear curve obtained from plotting a hodograph at different altitudes, obtained from sounding data.

With this information, the wind profile (Figure 3.7) is determined from the sounding and applied to the advection of a particle (see Algorithm 3).

3.5 SkewT/LogP Diagrams

Equation (3.8) is the core of our cloud simulator because it provides us a description of the upwards motion of any air parcel that evolves to a cloud. We do not use the Navier-Stokes equations in our formulation, nor any numerical integration scheme to solve



(a) 8UTC (b) 14UTC

Figure 3.7: Wind profile extracted from soundings related to 8 UTC and 14 UTC.

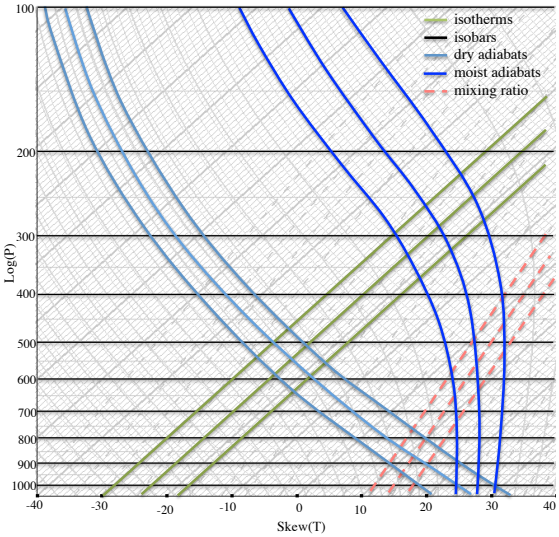
Equation (3.8). Solving this motion equation for each particle is *explicitly* accomplished using Equation (3.7), which in turn requires the values of two parameters, temperature T and pressure P. As explained below, this pair of parameters is extracted from a SkewT/LogP diagram.

Note that a SkewT/LogP diagram [AWSH61] is a meteorological tool to observe weather elements at every pressure plane in the atmosphere, being that each pressure plane is represented by an isobar line, that is, an horizontal line (in black) of equal pressure (see Figure 3.8(a)).

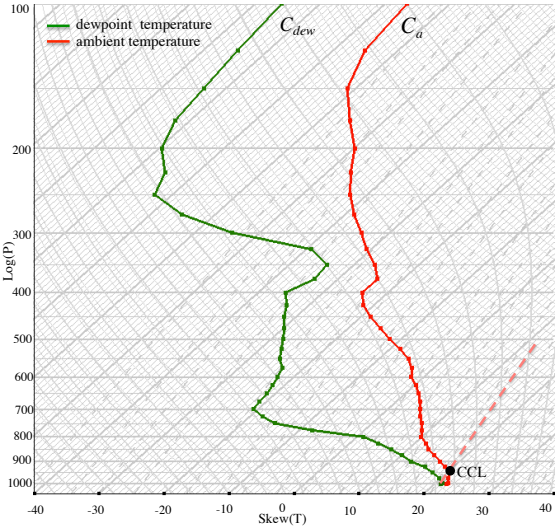
Moreover, a SkewT/LogP diagram allows us not only to determine the temperature and pressure of each cloud particle in the atmosphere, but also the three important parameters that feature the three stages of formation of clouds (cf. Figure 3.8), namely: convective temperature (T_c), convective condensation level (CCL), and equilibrium level (EL). In fact, using a SkewT/LogP, all parameters required to solve Equation (3.8) are determined geometrically by intersecting curves in the SkewT/LogP diagram with data acquired from the sounding, as shown in Figure 3.8.

3.5.1 SkewT/LogP Curves

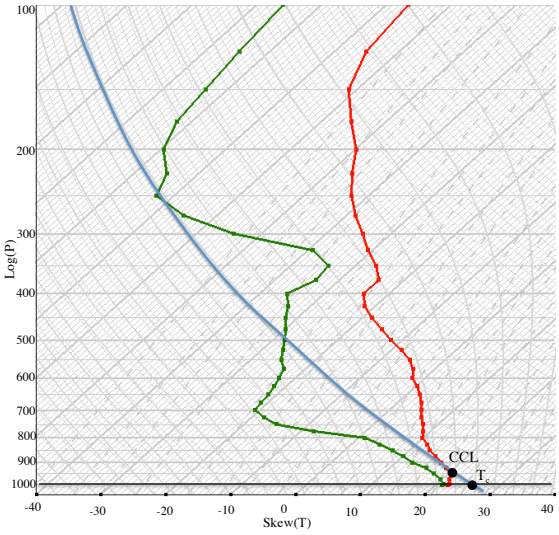
A SkewT/LogP diagram allows us to determine a number of parameters that are necessary to simulate the upwards motion of air parcels. So, before proceeding any further, let us state the fundamental curves (or lines) of a SkewT/LogP diagram we need to later calculate those parameters. As shown in Figure 3.8(a), there are five curves that represent standard properties of air parcels, namely: *isobars*, *isotherms*, *mixing ratio lines*, *dry adiabats*, and *moist adiabats*.



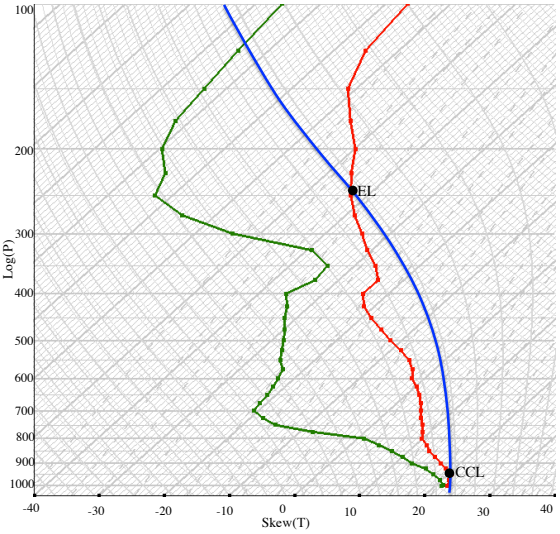
(a) Highlight of the background curves.



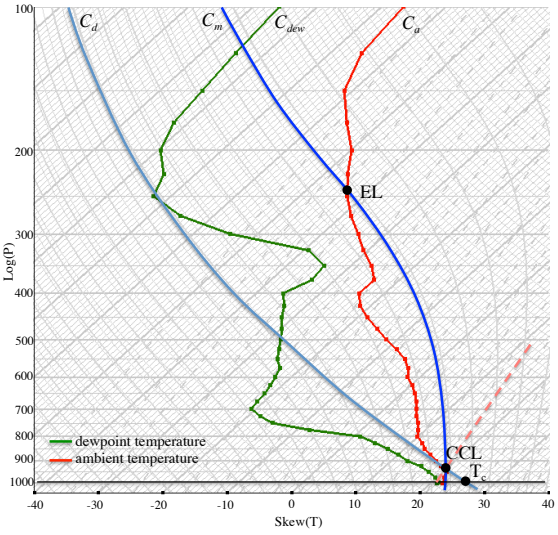
(b) Geometrical procedure to estimate the CCL.



(c) Geometrical procedure to estimate T_c.



(d) Geometrical procedure to estimate the EL.



(e) All parameters together.

Figure 3.8: SkewT/LogP diagram representation and estimation of convective cloud formation parameters. The vertical scale is LogP and labeled values are given in terms of pressure P.

3.5.1.1 Isobars

Isobars are horizontal lines (in black) in Figure 3.8(a). An isobar represents diagram points of constant pressure. In a SkewT/LogP diagram, they are traced every 50 mb in a vertical LogP scale. The pressure P=1000 mb at the origin of the SkewT/LogP chart corresponds to the sea level pressure.

3.5.1.2 Isotherms

Isotherms are skewed straight lines (in mossy green) that are traced from the lower left to the upper right in Figure 3.8(a). They are 45-degree rising lines parallel to the line $y = x$ (i.e., 45-degree slope) in Cartesian coordinates. An isotherm features diagram points with the same temperature, which typically is in the interval $[T_{\text{min}}, T_{\text{max}}]$ and the value of the Cartesian coordinate x in $[x_{\text{min}}, x_{\text{max}}]$. Pressures in the interval $[P_{\text{min}}, P_{\text{max}}]$ are converted into a y-value in the interval $[y_{\text{min}}, y_{\text{max}}]$ that decreases as pressure increases. The minimum pressure is mapped into the y_{max} value and the highest pressure is mapped into y_{min}. Isotherms can be then calculated using the following coordinate transformation:

$$\begin{aligned} x &= \frac{x_{\text{max}} \cdot (T - T_{\text{min}})}{T_{\text{max}} - T_{\text{min}}} + y - x_{\text{max}}/2 \\ y &= y_{\text{max}} - \frac{y_{\text{max}} \cdot (\log_{10}(P) - log_{10}(P_{\text{min}}))}{\log_{10}(P_{\text{max}}) - \log_{10}(P_{\text{min}})} \end{aligned} \tag{3.10}$$

That is, a point (T, P) in the SkewT/LogP chart maps onto a point (x, y) in Cartesian coordinates [oCoDoT72].

3.5.1.3 Mixing Ratio Lines

Mixing ratio lines (or isohumes) w are lines of equal mixing ratio that describe the saturation mixing ratio of air. They relate the mass of water vapour in a parcel (in g) to the mass of dry air (in kg) as follows (in g/kg):

$$w(T, P) = \frac{\varepsilon \,.\, e(T)}{P - e(T)} \tag{3.11}$$

where $\varepsilon = R_d/R_m$, and $e(T)$ stands for the saturation vapor pressure that can be approximated by August-Roche-Magnus formula [Law05], given by:

$$e(T) \approx C \exp\left(\frac{A \,.\, T}{T + B}\right) \tag{3.12}$$

where $A = 17.625$, $B = 243.04$ and $C = 610.94$ [AE96].

Given a constant value for the mixing ratio $w(T, P) = W$, we can express T in terms of

P by rearranging the expression of Equation (3.11) as follows:

$$T(P) = \frac{B \ln\left(\frac{W \cdot P}{C(W+\varepsilon)}\right)}{A - \ln\left(\frac{W \cdot P}{C(W+\varepsilon)}\right)} \tag{3.13}$$

We use Equation (3.13) to determine the mixing ratio line that passes through (T, P) by calculating the value of the temperature $T(P+\delta)$, with δ being a small integer. The points (T, P) and $(T(P+\delta), P+\delta)$ define a mixing ratio line, whose points have all the same mixing ratio.

In a SkewT/LogP diagram, mixing ratio lines are plotted as dashed near-straight curves (in brown) from the lower left to the upper right, as shown in Figure 3.8(a). As explained further ahead, mixing ratios are important to determine the height at which condensation occurs, that is, the convective condensation level (CCL), as illustrated in Figure 3.8(b).

3.5.1.4 Dry Adiabats

Dry adiabats are the navy blue curves in the SkewT/LogP diagram shown in Figure 3.8(a). They feature the thermodynamic behavior of unsaturated air parcels moving upwards (or downwards), that is, they represent the dry adiabatic lapse rate (DALR). This thermodynamic behavior is valid for all air parcels moving between the ground and the convective condensation level (CCL). The dry adiabat curves are described by the following expression:

$$T(P) = \theta \Big/ \left(\frac{P_0}{P}\right)^{R_d/c_p} \tag{3.14}$$

after rearranging T in terms of P in Equation (3.7), where P_0 is the initial value of pressure (i.e., ground pressure). Thus, a dry adiabat is a curve whose points contain the same potential temperature θ. That is, every single dry adiabatic curve corresponds to a single temperature θ on the temperature axis; hence, dry adiabatic curves are 1/logP curves. We use Equation (3.14) to draw the dry adiabat curve with the potential temperature θ, computing the temperature T for successive values of pressure $P \in [P_0, P]$.

As shown further ahead, the dry adiabat curve (3.14) is necessary to calculate the convective temperature of particles of an air parcel on the ground. But, more importantly, the dry adiabats describe the thermodynamic behavior of particles during their ascending motion from the ground up to the CCL (see Figure 3.8(c)).

3.5.1.5 Moist Adiabats

Moist adiabats, also called pseudoadiabats, are the blue curves in Figure 3.8(a) that run from the bottom of the diagram, curving then gradually toward the upper left, so that ultimately they become nearly parallel to the dry adiabats. They represent the saturated adiabatic lapse rate (SALR). The AMS *Glossary of Meteorology* provides a detailed description of the formula for the lapse rate of the pseudo-adiabatic process as follows:

$$\Gamma(T,P) = g\frac{(1+w(T,P))\left(1+\frac{L_v(T)w(T,P)}{R_dT}\right)}{c_{pd}+w(T,P)c_{pv}+\frac{L_v(T)^2w(T,P)(\varepsilon+w(T,P))}{R_dT^2}} \tag{3.15}$$

where $w(T,P)$ is the mixing ratio of water vapour, c_{pd} and c_{pv} are the specific heat of dry air and the specific heat of water vapour, respectively, at constant pressure, and $L_v(T)$ stands for the latent heat of vaporisation/condensation given by

$$L_v(T) = \left(aT^3+bT^2+cT+d\right).1000 \tag{3.16}$$

where $a = -6.14342\times10^{-5}$, $b = 1.58927\times10^{-3}$, $c = -2.36418$ and $d = 2500.79$ (cf. [RY89]). The pressure lapse rate is given by $dT/dP = \Gamma(T,P)/\rho g$, so by iintegrating this expression from a starting pressure P_0 and temperature $T(P_0)$, we obtain the new temperature $T(P_1)$ at the new pressure P_1 as follows:

$$T(P_1) = T(P_0) + \int_{P_0}^{P_1}\frac{\Gamma(T,p)}{\rho g}dp \tag{3.17}$$

where the density ρ is given from the ideal gas law as follows:

$$\rho = \frac{P-e(T)}{R_dT}+\frac{e(T)}{R_mT} \tag{3.18}$$

Summing up, the mathematical expressions of the above five thermodynamic curves (or lines) enable us to draw SkewT/LogP diagrams, as illustrated in Figure 3.8(a). In particular, the mixing ratio lines, dry adiabats, and moist adiabats are very important because they provide information about upwards motion (i.e., convection) of air parcels in the atmosphere.

3.5.2 Sounding Curves

In addition to those five types of thermodynamic curves (or lines) in Figure 3.8(a), we need two more curves (called sounding curves), represented by the piece-wise linear approximations in Figure 3.8(b), to simulate the upwards motion of air parcels or, equivalently, to solve the cloud motion equation (3.8). The first (red) of these two curves features the ambient temperature T_a, whereas the second (green) concerns dew-point

temperature T_d. Interestingly, in [MGG+10], T_a and T_d are also used to characterize the evolution of weather in an environmental model of snow accumulation; more specifically, these curves are used to determine the likelihood of falling snow during the day.

In our work, these sounding curves are generated from atmospheric soundings taken from meteorological agencies around the world (cf. Section 3.1). Recall that an atmospheric sounding is a set of data collected by a balloon (with attached sensor and transmission equipment known as radiosonde), which moves upwards in the atmosphere. Atmospheric soundings are used to diagnose and predict the potential of convective activity and, ultimately, to forecast weather over the planet (Figure 3.9).

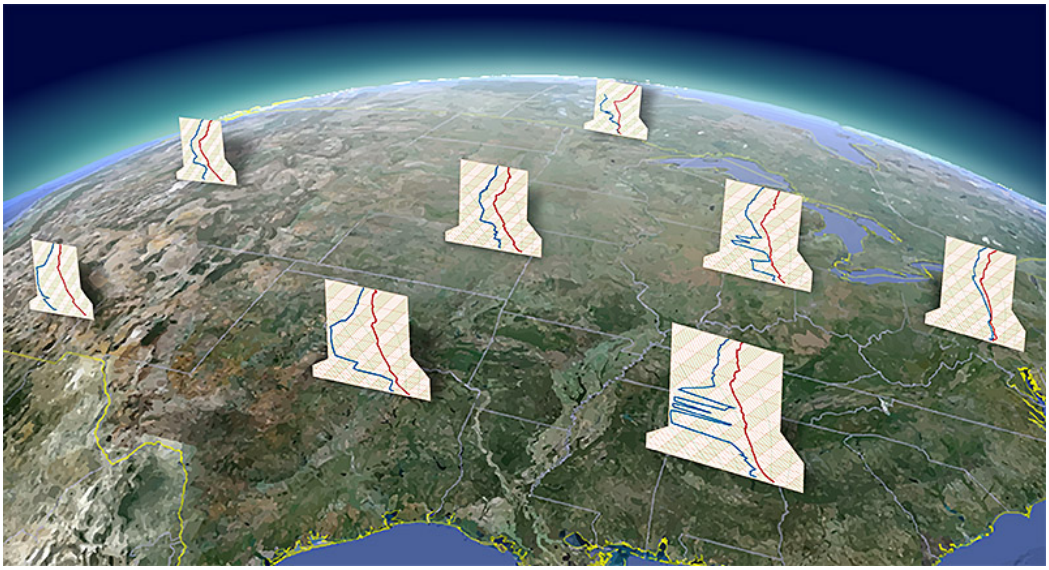


Figure 3.9: Upper air observations from a few out of 90 observations sites in the United States. National Weather Service

As illustrated in Table. 3.1, for raw data collected at different pressure levels, different parameters are considered such as the pressure level of the balloon PRES (in hPa), height HGHT (in meters), ambient temperature TEMP (in °C), dew-point temperature DWP (in °C), relative humidity RELH (in %) of the air parcel, mixing ratio MIXR (in g/kg), wind direction DRCT (in °) given by the angle of the wind relative to the balloon position, and wind speed SKNT (in knots). The sounding curves for the ambient temperature T_a and the dew-point temperature T_d are specifically generated from the data listed in the third and fourth columns shown in Table. 3.1, respectively.

The leading idea of our method is thus to use these meteorological parameters to *explicitly* solve the upwards motion equation (3.8) in order to realistically simulate and render the formation of clouds.

Table 3.1: An example of an atmospheric sounding in tabular format extracted from http://www.twisterdata.com/, for a convective region.

PRES (hPa)	HGHT (m)	TEMP (°C)	DWPT (°C)	RELH (%)	MIXR (g/Kg)	DRCT (deg)	SKNT (knot)	TWTB (°C)	TVRT (°C)	THTA (K)	THTE (K)	THTV (K)
1004.6	3	22.2	20.9	91	15.7	173	12	21.4	25.0	295.0	340.2	297.8
1000.0	39	22.8	21.1	90	16.0	168	20	21.7	25.6	295.9	342.2	298.7
975.0	260	21.6	20.2	91	15.5	170	31	20.7	24.3	296.8	341.8	299.6
950.0	486	20.4	19.2	93	15.0	171	38	19.6	23.0	297.8	341.5	300.5
925.0	716	18.9	17.8	93	14.1	177	41	18.2	21.3	298.6	339.7	301.1
900.0	952	17.4	16.3	93	13.1	186	42	16.7	19.6	299.4	337.9	301.7
875.0	1193	15.8	14.9	94	12.3	194	43	15.2	17.9	300.1	336.4	302.3
850.0	1439	14.2	13.4	95	11.5	202	42	13.7	16.1	300.9	335.0	303.0
825.0	1691	12.9	11.5	91	10.4	206	40	12.0	14.6	302.1	333.2	304.0
800.0	1950	12.0	8.5	79	8.8	209	38	9.8	13.5	303.8	330.3	305.5
775.0	2215	11.0	2.6	56	6.0	210	36	6.3	12.0	305.5	324.0	306.6
750.0	2488	9.8	-2.0	43	4.4	211	34	3.7	10.5	307.1	321.1	307.9
...	...	...	...	...	...	...	...	...	...	...	...	...
100.0	16505	-68.6	-87.2	5	0.0	212	40	-69.0	-68.6	394.6	394.6	394.6

3.6 Cloud Formation

In a particle system, each particle crosses three consecutive stages: birth, life and death. Cloud formation comprises birth and the first part of life, while cloud dissipation refers to the second part of life and death.

More specifically, cloud formation has three distinct steps: *birth*, *dry lift*, and *moist lift*, as presented in Figure 3.10. These two latter steps are part of cloud life.

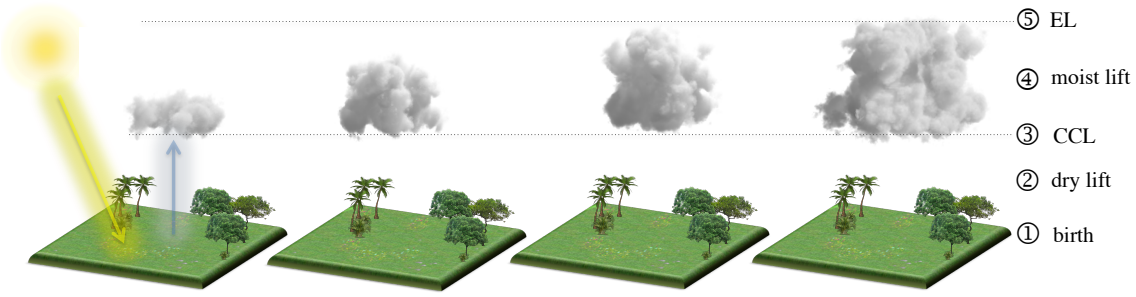


Figure 3.10: Overview of the cloud formation algorithm for a cloud with 5,000 particles: The birth of particles within a parcel of air occurs when they reach the T_c due to the warming of Earth's surface ①. These particles of dry air rise into the atmosphere ② at the dry adiabatic lapse rate (Section 3.6.2). When this parcel rises, it expands and cools. Water vapor condenses ③ out from the dry air at the CCL giving rise to a cloud. In the presence of moist air, the air parcel continues to rise (Section 3.6.3) at the moist adiabatic lapse rate ④, until equilibrium is reached ⑤.

3.6.1 Birth

The birth of particles includes gestation (or generation) and parturition of particles. During the gestation stage, one generates the particles of each air parcel while the sun warms the ground, i.e., particles end getting more and more warm by absorption of solar radiation. With this warming process, each particle ends up reaching its convective

temperature T_c at some point (i.e., parturition), which is the temperature that makes it starting to rise above the ground (i.e., P_0). Therefore, T_c has to be computed before anything else. This is summarised in Algorithm 1, which consists of the following five steps:

- *Step 1.* The dewpoint (T_{dew}, P_0) on the ground is the first pair of values of the dewpoint temperature curve C_{dew}. For example,looking at Table 3.1, we see that $T_{\text{dew}} = 20.9$°C (2nd column) and $P_0 = 1004.6$ hPa (1st column), with the ground located at 3 meters above the sea level.
- *Step 2.* The method described in Sec. 3.5.1.3 gives the mixing ratio line l that passes through the dewpoint (T_{dew}, P_0). In the case of the sounding data in Table 3.1, we would have to use the values $T_{\text{dew}} = 20.9$°C (4th column), $P_0 = 1004.6$ hPa (1st column), and $w = 15.7g/kg$ (6th column).
- *Step 3.* The CCL point $(T_{\text{CCL}}, P_{\text{CCL}})$ is the point that results from the intersection between the mixing ratio line l (see previous step) and one of the line segments of the piece-wise linear curve C_a relative to ambient temperature. Therefore, this reduces to the well-known line-to-line intersection problem.
- *Step 4.* The computation of the potential temperature θ at the CCL is given by Equation (3.7); more specifically, it is given by $\theta = T_{\text{CCL}}(\frac{P_0}{P_{\text{CCL}}})^{R_d/c_p}$, where $(T_{\text{CCL}}, P_{\text{CCL}})$ stand for the temperature and pressure at CCL.
- *Step 5.* The potential temperature θ of a particle is constant along the dry adiabat given by Equation (3.7), so that we can use the value of the potential temperature determined before to calculate the convective temperature T_c (shown in Figure 3.8(c)) of each particle on the ground, that is, at the point (T_0, P_0), being $T_0 = T_c$. Therefore, $\theta = T_0(\frac{P_0}{P_0})^{R_d/c_p}$, that is, the convective temperature is given by $T_c = T_0 = \theta$. Note that T_c is greater than the ambient temperature at the same level of pressure.

ALGORITHM 1: Computation of the convective temperature T_c for an air parcel.

begin

$(T_{\text{dew}}, P_0) \leftarrow$ dewpoint on the ground	◁ Table 3.1
$l \leftarrow$ mixing ratio line through (T_{dew}, P_0)	◁ Equation 3.13
$(T_{\text{CCL}}, P_{\text{CCL}}) \leftarrow l \cap C_a$	
$C_d \leftarrow$ dry adiabat curve through $(T_{\text{CCL}}, P_{\text{CCL}})$	◁ Equation 3.14
$T_c \leftarrow C_d \cap P_0$	◁ intersection between C_d and isobar P_0

It should be said that particles of air parcels are generated randomly by one or more emitters located on the ground. For each particle, a set of behaviour attributes is assigned, namely: position, convective temperature (T_c) −also known as emission temperature−, condensation (CCL), and equilibrium (EL).

Let us then consider the first two attributes of particles, since the others will be discussed in the following sections. The position of each particle is generated in a random

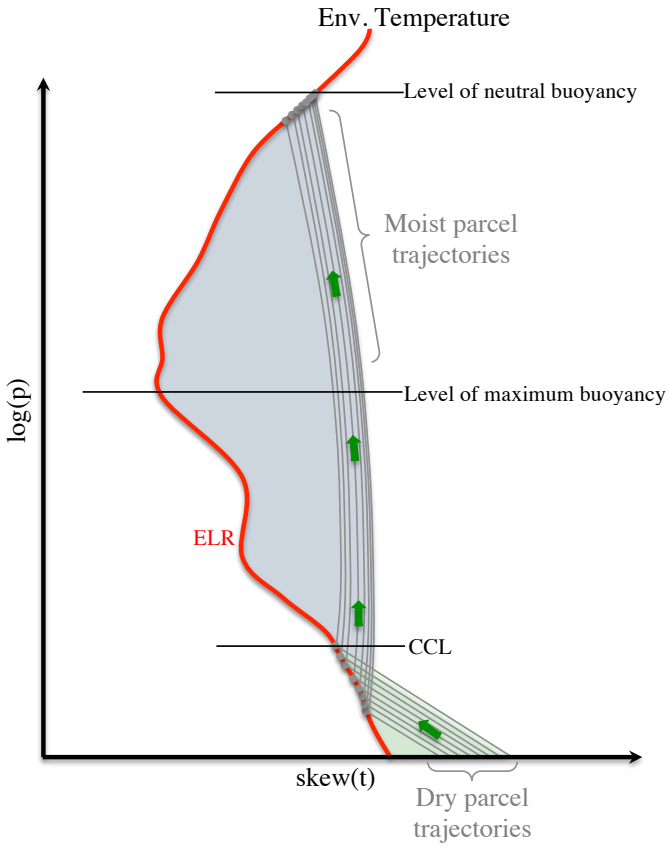


Figure 3.11: Particle trajectories (thin gray curves denote the C_d and C_m curves in Algorithm 3) versus the environment temperature curve (bold red line denotes curve C_a in Algorithm 3). The particle temperature trajectories are indicated as green arrows along dry and moist adiabats.

manner within the region occupied by 2D grid on the ground, which is related to scene's terrain. This on-ground grid can be delimitated by a circle, a rectangle, or even a free shaped frontier when it is seen from the top downwards, which has impact on the shape of the cloud. The position of each particle π_i is generated in a random manner within the region on the ground, which is related to the scene's terrain, and its position is given by $(x_i, y_i, z_i) = (x, y, z_0)$, where x and y are random numbers, z_0 is the height (in meters) of the particle at pressure P_0, being the velocity set to zero, $v_i = 0$. Also, the convective temperature of each particle is generated randomly within the interval $[T_c, T_c + \Delta t]$, where T_c is the convective temperature associated to the dry adiabat curve passing through (T_{CCL}, P_{CCL}). In this way, we end up having a dry adiabat curve for each particle because the potential temperature that characterizes each adiabat is equal to the convective temperature of the particle.

In practice, instead of creating a temperature path (i.e., an adiabat) for each particle, we predefine a small number of paths for a large number of particles (see Figure 3.11), reducing this way the overall computational burden of the simulation. For example, considering 80 paths and assuming that $T_c = 20\,^{\circ}C$ and $\Delta t = 1.0$, one out of 80 paths is generated at every $0.0125\,^{\circ}C$. Let us consider that one of those particles is generated with temperature $20.44\,^{\circ}C$. This particle, instead of having its own temperature path, it is assigned the nearest path (with closest temperature value), which corresponds to

the profile 35 (out of 80) with the temperature $20.4375\,^{\circ}C$.

After assigning a convective temperature to every single particle of each parcel in the birth stage, we are in position to proceed to the second stage, called dry lift of air parcels. The dry lift has to do with the particle rising from the ground to the convective condensation level (CCL). Before reaching the CCL, particles are hotter than the environment surrounding them (due to the energy gain in the first stage), but they cool down as they go up. This means that particles continue to rise and cool while their temperatures are higher than the surrounding temperature.

Therefore, we need to calculate the CCL for each particle π_i of a given parcel $\mathbf{\Pi}$, as described in Algorithm 2. Note that the CCL of a given particle π_i is the point (T_i, P_i) that results from the intersection between its dry adiabat curve and the piece-wise linear curve C_a interpolating the sounding ambient temperatures T_a (cf. 3rd column of Table. 3.1). Thus, we first calculate the potential temperature $\boldsymbol{\theta}(T_i, P_i)$ at every single vertex of the C_a, as illustrated by the first 'for' loop in Algorithm 2. This is equivalent to determine the dry adiabat that goes through each vertex of C_a, so we end up having a sequence $\boldsymbol{\theta}_0, \ldots, \boldsymbol{\theta}_n$ of dry adiabats (or potential temperatures) crossing C_a on its vertices.

ALGORITHM 2: FindCCLforEachParticleofParcel$(\Pi, \{T_{c_i}\}, \mathcal{D})$

Data: Π ◁ air parcel Π
Data: $\{T_{c_i}\}$ ◁ convective temperatures for particles $\{\pi_i\}$ of Π
Data: $\mathcal{D}$ ◁ sounding data in Table 3.1
Result: $\{\theta_j\}$ ◁ potential temperatures $\{\theta_j\}$ for nodes of C_a
Result: $\{(T_{\mathsf{CCL}_i}, P_{\mathsf{CCL}_i})\}$ ◁ CCL points for particles $\{\pi_i\}$ of Π
begin
  $k \leftarrow 0.286$ ◁ see Equation (3.6)
  $P_0 \leftarrow$ pressure on the ground ◁1st column in Table 3.1
  foreach *j-th row of $\mathcal{D}$* **do**
    $P \leftarrow j$-th pressure of $\mathcal{D}$ ◁ 1st column in Table 3.1
    $T \leftarrow j$-th temperature of $\mathcal{D}$ ◁ 3rd column in Table 3.1
    $\theta_j \leftarrow T\left(\frac{P_0}{P}\right)^k$ ◁ potential temperature, Equation (3.7)
  foreach $\pi_i \in \mathbf{\Pi}$ **do**
    $[\theta_j, \theta_{j+1}] \leftarrow$ determine interval: $T_{c_i} \in [\theta_j, \theta_{j+1}]$;
    $(T_{\mathsf{CCL}_i}, P_{\mathsf{CCL}_i}) \leftarrow$ find CCL: $\theta(T_{\mathsf{CCL}_i}, P_{\mathsf{CCL}_i}) = T_{c_i}$

Taking into account that the convective temperature T_{c_i} of a particle π_i coincides with the potential temperature of its dry adiabat, we have then to determine the interval $[\theta_j, \theta_{j+1}]$ where T_{c_i} fits in. To identify the segment $(j, j+1)$ of the C_a that crosses the dry adiabat of the particle π_i, we use a numerical zero finder to determine the CCL point $(T_{\mathsf{CCL}_i}, P_{\mathsf{CCL}_i})$ such that $\theta(T_{\mathsf{CCL}_i}, P_{\mathsf{CCL}_i}) = T_{c_i}$ (cf. Equation (3.7)), as can be observed in the second 'for' loop of Algorithm 2. During the dry lift, the particle position is updated with regard to the thermodynamic path assigned to each particle on the ground. This path is regulated by the dry adiabat curve while the particle does not attain the CCL, after which the thermodynamic behavior of the particle becomes regulated by the corresponding moist adiabat curve. That is, the entire lift motion (i.e., dry lift and

moist lift) of each particle is ruled by Equation (3.8).

3.6.2 Dry Lift

To solve Equation (3.8) at a point (T, P) located at altitude z, we need to know two values, θ_{π_i} and θ_a. The value of potential temperature θ_{π_i} of each particle is constant and is known from the previous step, and is equal to the convective temperature of such particle. Thus, it only remains to determine the ambient temperature θ_a (on the ambient temperature curve C_a) at the same level of pressure P. This computation consists of finding the point (θ_a, P) that stems from the intersection between the isobar P corresponding to altitude z and a segment of the curve C_a, which reduces to a line-to-line intersection problem. It is then necessary to calculate the rising force F to advect an air parcel, as well as its velocity v and the corresponding vertical displacement Δz, in order to determine the new position $z + \Delta z$ of each particle, and the particle evolution in (x, y) direction by including wind, as shown in Algorithm 3.

ALGORITHM 3: DryLiftMotion$(\mathbf{\Pi}, C_a, \{T_{c_i}\})$

Data: $\mathbf{\Pi}$ ◁ air parcel $\mathbf{\Pi}$
Data: C_a ◁ ambient temperature curve for parcel
Data: $\{T_{c_i}\}$ ◁ convective temperatures for particles $\{\pi_i\}$ of $\mathbf{\Pi}$
begin
  $\mathcal{N} \leftarrow$ number of particles in the air parcel Π
  count $\leftarrow 0$ ◁ number of particles at the CCL
  while count $< \mathcal{N}$ **do**
    foreach $\pi_i \in \mathbf{\Pi}$ **do**
      if $P_i > P_{CCL_i}$ **then**
        $l \leftarrow$ isobar line at pressure P_i
        $C_d \leftarrow$ dry adiabat for π_i, $\theta = T_{c_i}$ ◁ Equation (3.14)
        $(T_A, P_A) \leftarrow (l \cap C_a)$
        $(T_B, P_B) \leftarrow (l \cap C_d)$
        $\theta_a \leftarrow$ potential temp. at (T_A, P_A) ◁ Equation (3.7)
        $\theta_i \leftarrow$ potential temp. at (T_B, P_B) ◁ Equation (3.7)
        $a \leftarrow 9.81 \cdot (\theta_i - \theta_a)/\theta_a$ ◁ Equation (3.8)
        $v_i \leftarrow v_i + a \cdot t$ ◁ velocity of i-th particle
        $\Delta z \leftarrow v_i \cdot t + 1/2 \cdot a \cdot t^2$ ◁ truncated Taylor series
        $z_i \leftarrow z_i + \Delta z$ ◁ altitude of i-th particle
        $x_i \leftarrow x_i + \Delta x$ ◁ wind Δx from Equation (3.9)
        $y_i \leftarrow y_i + \Delta y$ ◁ wind Δy from Equation (3.9)
        $P_i \leftarrow (\frac{44331.514 - z_i}{11880.516})^{1/0.1902632}$ ◁ pressure at z_i
      if $P_i \leq P_{EL_i}$ **then** count = count +1;

3.6.3 Moist Lift

At the third stage, particles advect from the CCL to the EL. Particles may advect positively or negatively, depending on whether the environment temperature is higher or lower than the particle temperature (given by a moist adiabat). When they reach the

EL, they oscillate around this level (buoyancy effect) and tend to become stationary inside the cloud.

Similarly to the dry lift, in the moist lift we have to determine both the EL and the force required to advect particles between the CCL and the EL. The algorithm to determine the EL for each particle is similar to Algorithm 2, with the difference that, instead of determining the temperature for a dry adiabat curve, one determines the temperature for a moist adiabat curve that passes through the CCL and intersects again the ambient curve, as presented in Figure 3.8(d). For the moist lift, the convection process is similar to the one presented in Algorithm 3 with the difference that, C_d is the moist adiabat curve that passes through the EL, C_m.

3.7 Results

Our results were obtained on a desktop computer with and Intel Core i7 with 3.07 GHz, running Windows operating system (v7) with 24 GB of RAM, and an Nvidia Quadro 6000 6GB graphics card.

3.7.1 2D SkewT/LogP Simulator

Our 2D SkewT/LogP simulator shown in Figure 3.12 is the core of our work since it provides information (curves) to determine the temperature profile of particles in the cloud. All the background curves from the SkewT/LogP diagram and the temperature curves —obtained from data— are stored in a data structure used to determine cloud levels and to determine the vertical motion of particles, by calculating the intersection points between these curves, as referred in previous sections. Furthermore, it allows us to control the upward motion of clouds in the atmosphere even when the meteorological conditions change over time, i.e., from one atmospheric sounding to another. A detailed explanation of Figure 3.12 is given in Appendix D.

The user can also interactively change temperature curves (C_a and C_{dew}) directly in the 2D interface —in order to obtain clouds with desired shapes— by picking up and sliding the data points of these curves along each pressure level in the SkewT/LogP diagram, and then all parameters are recalculated. Moreover, users can feed the 2D interface with their own temperature curves.

Therefore, the SkewT/LogP diagram controls the behavior of particles in the z direction, while the wind profile controls the (x, y) directions. From the meteorology point of view, our system is even able to predict the occurrence of thunderstorms and stability indexes, but this is out of the scope of this work.

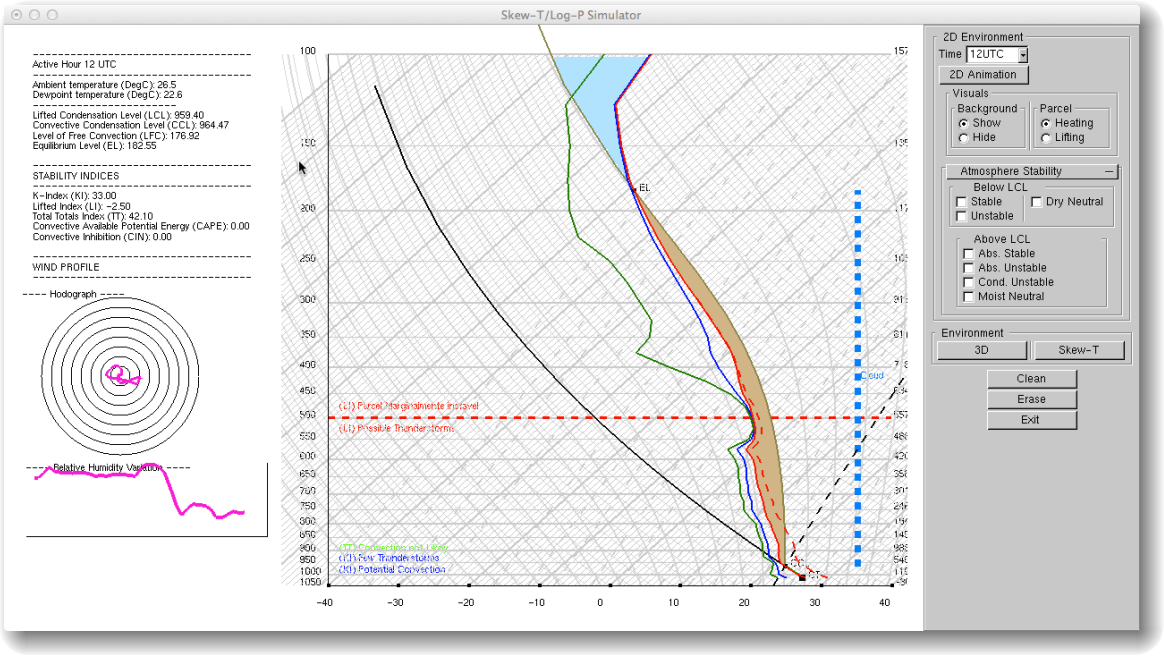
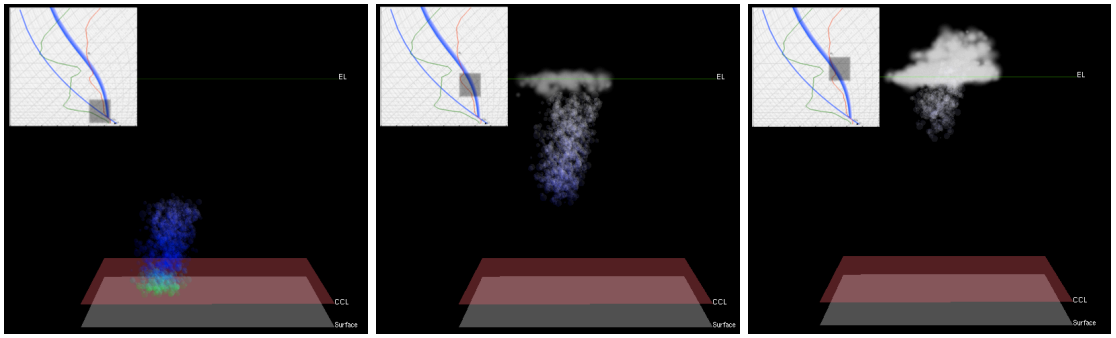


Figure 3.12: Our 2D user interface with the SkewT/LogP simulator for the estimation of temperature profiles based on sounding data. The represented sounding was obtained for the region of Virgin Islands, at 12 pm of 14/08/2013, extracted from http://www.twisterdata.com.

3.7.2 3D Cloud Simulation and Rendering

In order to simulate the formation of clouds in the atmosphere we have also developed a 3D particle visualizer that was implemented in C++ using the OpenGL/GLUT libraries, as presented in Figure 3.13 and Figure 3.14.



(a) (b) (c)

Figure 3.13: The influence of the wind on the shape of an air parcel/ cloud with 5000 particles: (a)-(c) snapshots of the air parcel/cloud where the horizontal deviation of particles caused by the wind are visible.

This simulator uses the data structure defined for the 2D simulator to solve the equation of motion in real-time for particles generated at ground level until equilibrium is not reached within the cloud. In Figure 3.13, a cloud is generated with wind acting on the air mass. The wind profile is presented in Figure 3.7 and is extracted from the sounding data, as referred in Section 3.4. The relation between the air mass ascension and

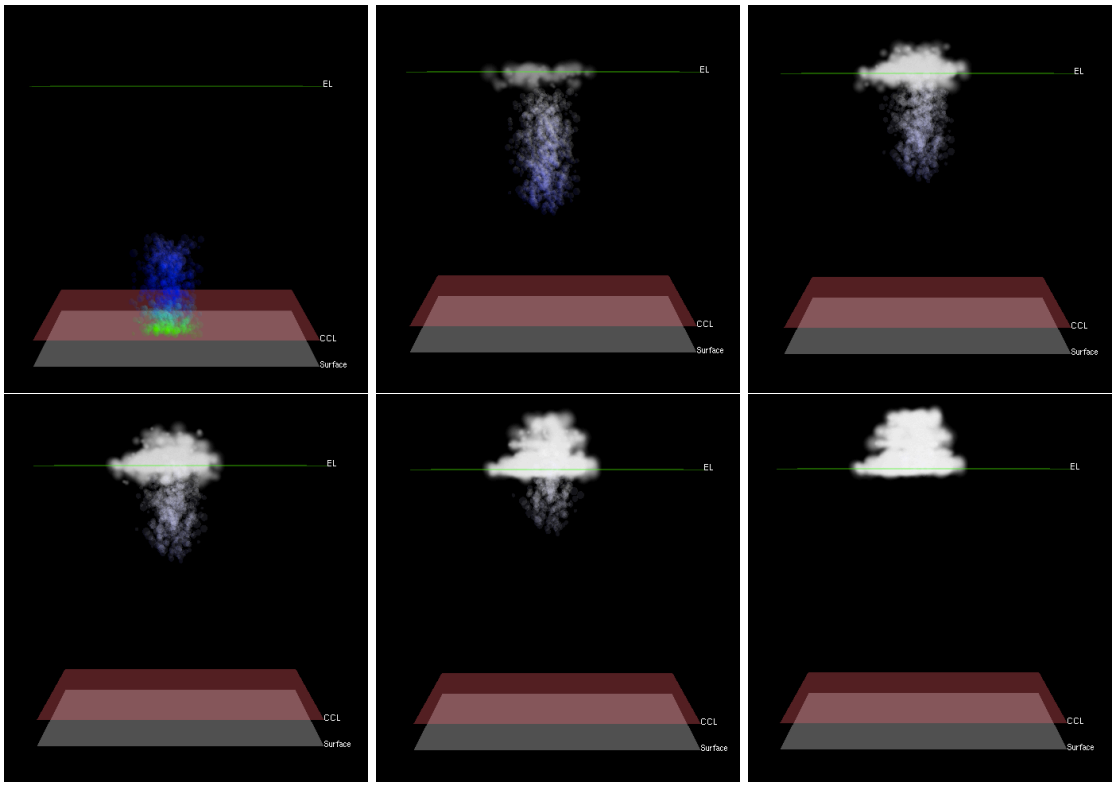


Figure 3.14: Different stages of the evolution of a cloud generated with 5000 particles without wind effect.

the SkewT/LogP diagram is also presented. Recall that particles only become visible after they reach their own CCL, and become stationary around the EL. For illustration purposes, we vary the colour of particles to show that the temperature of particles varies as clouds move vertically in the atmosphere. In Figure 3.14, clouds are generated at ground level and evolve in the atmosphere without the influence of wind.

To realistically render clouds, we have used Elementacular plugin [Ale15] for Maya software. This real-time rendering technique is based on three main steps as follows. First, a 3D mesh is created from spheres representing particle positions and densities generated by our 3D simulator, and a rasterization-based voxelization technique is applied to the mesh [CG12]. Second, three fields are computed to the generated voxels: a distance field, a fractal noise field and a wispy field based on gridless advection [Wre12]. Finally, illumination is applied to the cloud using a ray-marching technique for multiple scattering of light in clouds [KPS+14] (see Figure 3.15).

Let us mention that we have implemented other techniques for the realistic rendering of clouds [HL01] [HL02]. However, the visual results were not satisfactory; hence the choice of Maya's Elementacular plugin. Nevertheless, the focus of this thesis is on the simulation of cloud formation. The coupling of the Elementacular plugin to our 3D cloud simulator is illustrated in Figure 3.15. Essentially, the particle positions of each cloud are exported to Maya in each time step, so that the Elementacular plugin is responsible for rendering clouds in a realistic manner over time.

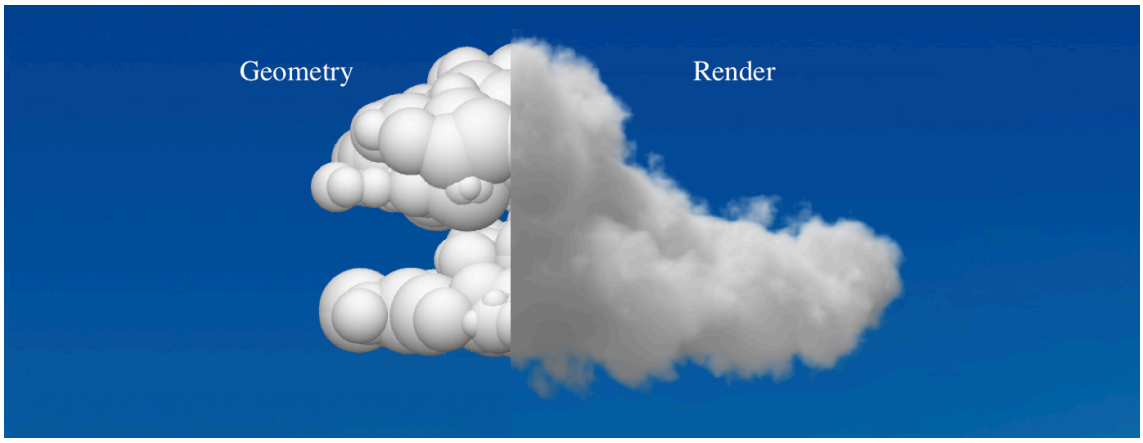


Figure 3.15: Illustration of the simulation and rendering: (left) geometry of a cloud generated using our simulator; (right) final result after the application of the rendering algorithm.

3.7.3 Cloud Shape

Several factors determine the overall shape of clouds: *sounding data*, *soil temperature*, *horizontal winds*, and *particle emission area*.

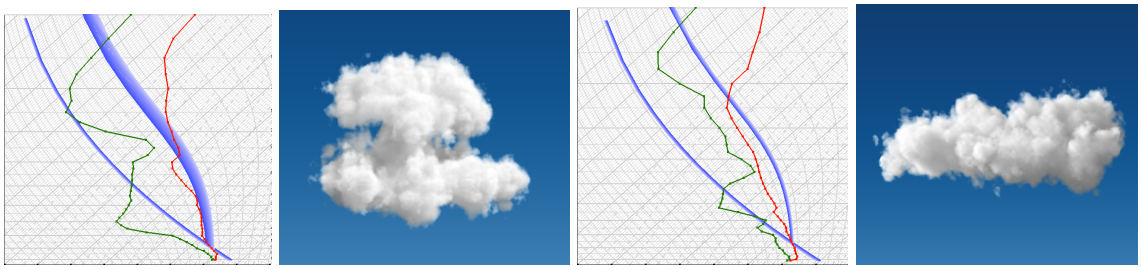
3.7.3.1 Sounding Data

The shape of a cloud essentially depends on its (vertical) *EL spread*, which can be defined as the height difference between the pressure levels concerning the minimum and maximum ELs in C_a for all particles of such cloud. These minimum and maximum ELs set the base and top of the cloud, respectively. The EL spread results from the intersection between the ambient temperature curve C_a and the moist adiabats, while the *CCL spread* results from the intersection between the ambient temperature curve C_a and the dry adiabats.

In Figure 3.16, we see two clouds generated from different soundings. We readily observe that the CCL spread for the cloud depicted in Fig. 3.16(a) is longer than the one shown in Fig. 3.16(b); as a consequence, the corresponding EL spread on C_a is also longer for the cloud on the left hand side than for the one on the right hand side. This shows the impact of the shape of C_a on the size of the CCL spread, which in turn determines the size of the EL spread. In fact, the EL spread determines whether the cloud grows deeper into the atmosphere (Fig. 3.16(a)) or not (Fig. 3.16(b)); the longer the EL spread on C_a, the taller is the cloud.

3.7.3.2 Soil Temperature

The second factor that influences cloud shape is the temperature profile associated to each particle inside the cloud. Depending on the temperature on the ground, particles inside clouds reach their equilibrium at a higher or lower altitude in the atmosphere.



(a) (b)

Figure 3.16: The influence of the sounding data on the shape of two clouds with 5,000 particles each. The convective temperatures of the particles of both clouds are all within equally-sized ranges in order to guarantee that the temperature does not interfere with the shape of clouds: (a) the cloud is expected to grow deep into the atmosphere because its CCL spread is broad, i.e., the moist adiabats form a broad set of curves; the convective temperature is $27.5°C$, with the ground temperature varying in the range $[27.5°C, 29.0°C]$; (b) on the contrary, the cloud is not expected to grow so deep into the atmosphere because its CCL spread is narrow, i.e., the moist adiabats form a narrow set of curves; the convective temperature is $25.2°C$, but the ground temperature varies in the range $[25.2°C, 26.7°C]$.

Assuming that particles are generated at ground level with temperatures within the interval $[a, b]$, one can simulate the variation of temperature during the day. We have studied three temperature profiles:

- *Increasing-temperature profile*. If the temperature always monotonically increases during a long period of the day, the ground temperature will also increase in the interval $[a, b]$. In this case, as illustrated in Fig. 3.17(a), clouds will be generated with turret shapes, but this also depends on the particle release pattern over $[a, b]$. In fact, the turret was formed by releasing more particles in lower and higher temperatures than in intermediate temperatures of the interval $[a, b]$. Note that lower-temperature particles originate the top of the cloud, while higher-temperature particles originate the bottom of the cloud, what is in conformity with the behaviour of dry and moist adiabats in the SkewT/LogP diagram.

- *Decreasing-temperature profile*. If the temperature always monotonically decreases during a long period of the day, the ground temperature decreases accordingly in the interval $[a, b]$. As shown in Fig. 3.17(b), the result is a cloud with an upside-down turret shape, but that also depends on the particle release pattern over $[a, b]$. In this case, about half of particles were released in higher temperatures, which led to the formation of the cloud base; the remaining particles were uniformly released over the remaining temperatures of $[a, b]$, having originated the uniform shape of the cloud top.

- *Mixed-temperature profile*. But, if the temperature oscillates in short periods of the day, the cloud shape tends to spread horizontally, largely because particles will be more more uniformly released over $[a, b]$ (see Fig. 3.17(c)-(d)).

Summing up, the cloud shape heavily depends on the particle release pattern over $[a, b]$. This suggests that one can model clouds with distinct shapes using a variety of particle

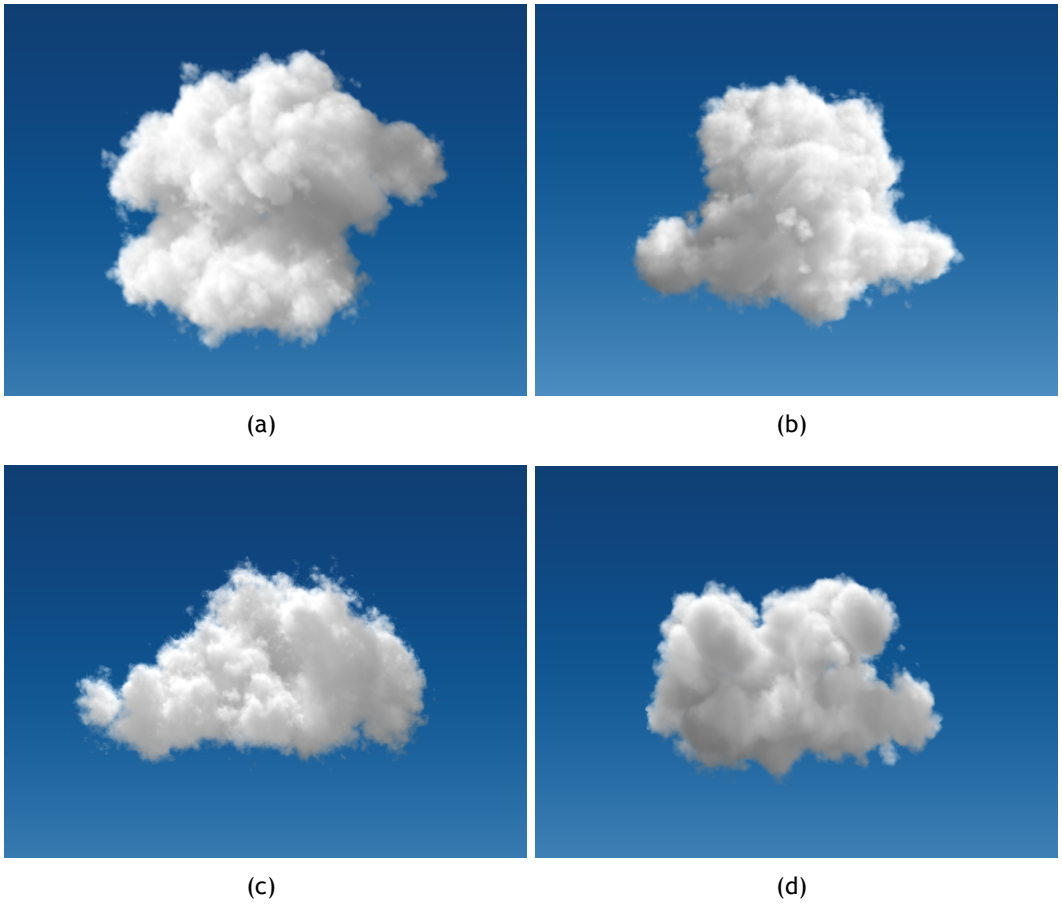
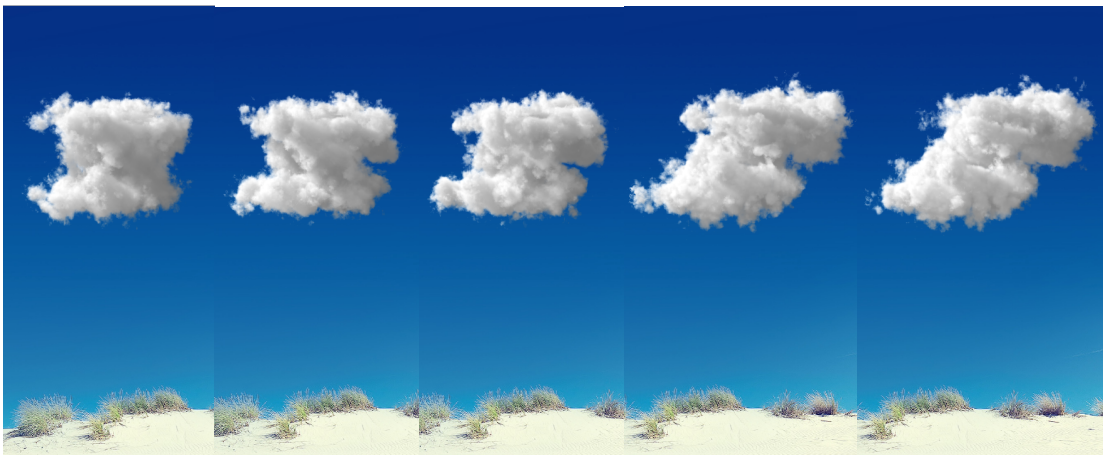


Figure 3.17: The influence of the soil temperature on the shape of a single cloud owing 5,000 particles in order to simulate diurnal variations of temperature, as regulated by the same sounding data: (a) the turret shape is achieved because the temperature monotonically increases during the day in the interval $T \in [24.63°, 25.93°]$, but more particles are released in lower and higher temperatures than in intermediate temperatures, being the equilibrium reached at higher altitudes for lower-temperature particles and at lower altitudes for higher-temperature particles; (b) the upside-down turret is achieved by monotonically decreasing the temperature during the day in the interval $T \in [25.83°, 26.13°]$, with half of the particles released in higher temperatures, and the remaining particles released uniformly with the steadily decreasing of temperature; (c) – (d) particles equally distributed for all temperature values in the interval $T \in [24.63°, 25.93°]$, but with the temperature increasing in a non-monotone manner, i.e., it increases but with small temperature decrease occurrences.

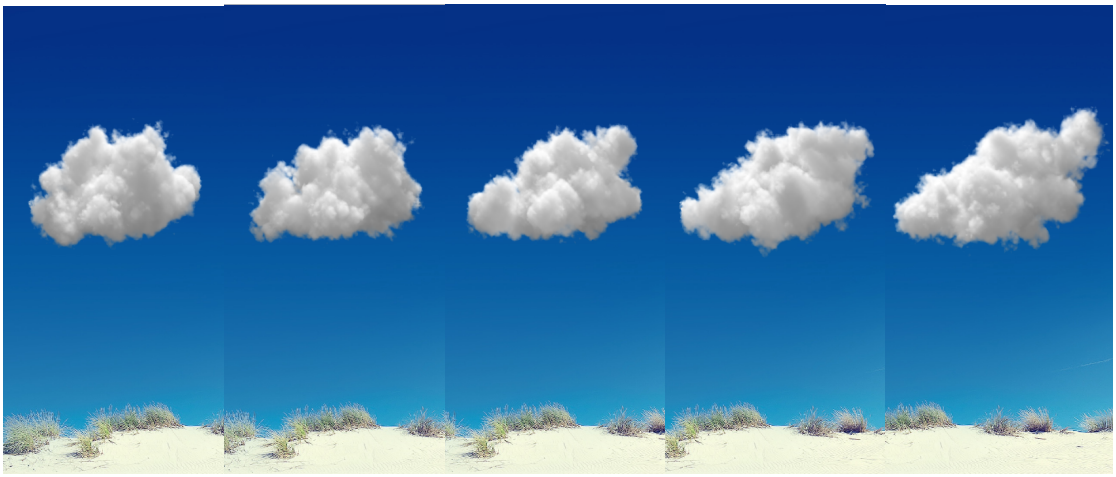
release patterns defined by real functions over $[a, b]$.

3.7.3.3 Horizontal Winds

The third factor with impact on the shape of clouds is wind. When horizontal wind blows against a cloud, its effect is more obvious at the top (rather than the bottom) of the cloud [Ema94]. This is illustrated in Figure 3.18 for two clouds composed of 15,000 particles. Its wind profile is plotted in Fig. 3.7(b), and was defined from the sounding data listed in Table 3.1 (the 7th column).



(a)



(b)

Figure 3.18: The influence of wind on the shape of a cloud with 15,000 particles: (a), (b) the sequence of snapshots shows the horizontal deviation of particles caused by the wind, which is more noticeable on the top of the cloud.

3.7.3.4 Particle Emission Area

The fourth factor that contributes to the shape of clouds is the area covered by particle emitters on the ground. The larger the area, the more expanded is the cloud or system

of clouds. This is illustrated in Fig. 3.19, where a number of clouds are spreading in the atmosphere. The clouds depicted in Fig. 3.19(a) were generated from particle emitters placed within an area of 1km $\times$ 1km on the ground, while those in Figs. 3.19(b)-(c) were generated from an area of 5km $\times$ 5km.

In short, sounding data and soil temperature determine the vertical expansion of clouds, whereas horizontal wind and placement of particle emitters determine their horizontal expansion in the atmosphere.

3.7.4 Interacting with clouds

We can change the shape of a cloud interactively, as well as its evolution in the atmosphere, as follows:

- By horizontally sliding one or more nodes (T, P) of dynamic curves C_a and C_{dew} along the same line of pressure P, we change the sounding data interactively; so, we change the dynamics of the cloud and its shape as a consequence of changing its EL spread (and also its CCL spread), as described in Section 3.7.3.1. A cloud gets taller when the difference between the minimum EL and maximum EL increases; otherwise, the cloud gets lower in height, though its width is more or less the same. Recall that different particles may have distinct CCLs and distinct ELs.
- Likewise, we can horizontally slide the ground node $(T, 1000)$ in the SkewT/LogP diagram of the 2D simulator in order to change the soil temperature that triggers the rising of air mass $-$that precedes the cloud$-$ in the atmosphere, as described in Section 3.7.3.2. In practice, this also changes the CCL and EL spreads. Consequently, this allows to control the height of a cloud, i.e., we are able to create taller/flatter clouds, yet its width remains the same approximately, as those depicted in Fig. 3.17.
- In order to generate large-scale clouds, we only have to populate a large ground area with particle emitters, as illustrated in Fig. 3.19.

However, as suggested above, the most promising solution to generate free-style clouds is to resort to particle releasing functions capable of characterizing the shape of clouds. To succeed on these free-style clouds, we have first to manage a way of converting a cloud sketch into a real function that regulates the particle releasing pattern for a cloud, but this is out of scope of this work.

3.7.5 Performance

Testing were performed in order to evaluate the efficiency of our algorithm in simulating and rendering clouds with increasing complexity. Two sorts of tests took place:

(a)

(b)

(c)

Figure 3.19: Cloud systems with increasing complexity composed of different clouds generated with soundings obtained on 20/05/2015. (a) Bottom view of a cloud system composed of 50,000 particles generated using a sounding for the region of Virgin Islands; (b) - (c) two different views of a cloud system composed of 1,000,000 particles generated using sounding data obtained for the region Manchester, United Kingdom.

- *Single cloud*. The first tests focused on the simulation and rendering of a single cloud.
- *Several Clouds*. The second tests involved the simulation and rendering of several clouds.

In practice, the number of clouds is irrelevant in terms of performance, because what matters is the total number of particles. It is noteworthy that we have only considered the time incurred in the physical simulation of the evolution of clouds in the atmosphere; the time performance does not include the rendering time.

3.7.5.1 Single Cloud

The first tests focused on the simulation of a single cloud. For each simulation step, particles inside a cloud are advected and rendered. As shown in Table 3.2, the generation of a realistic cloud in each simulation step takes about 3.08 milliseconds with wind acting on the air parcel or cloud composed of about 10,000 particles. When considering the time required to simulate the formation of a cloud (i.e., from the ground to EL), our simulator generates them in 0.212 minutes for a cloud composed of 10,000 particles (see Table 3.2). It happens that, in meteorology, clouds take about 5 to 45 minutes to be formed, so one concludes that we have real-time results on the cloud time scale.

Such real-time performance remains valid even for a cloud with 250,000 particles, because its formation in the atmosphere takes about 5 to 6 minutes. This mimics the reality provided that clouds are not formed instantly in real life. For a cloud with about 250,000 particles, our simulator takes about 44 milliseconds (52 milliseconds with wind) to process each cloud dynamics simulation step, i.e., what amounts to about 23 FPS (19 FPS if we consider the wind). Thus, the proposed method likely is suitable to simulate cloud shapes interactively, if we use CPU/GPU multi-threading or parallel processing.

# Particles	Average time step (miliseconds)		Time to generate a cloud (minutes)	
	with wind	without wind	with wind	without wind
10,345	3.08 ms	2.77 ms	0.212 min	0.195 min
32,327	4.46 ms	4.0 ms	0.52 min	0.43 min
64,535	6.67 ms	5.82 ms	1.04 min	0.85 min
96,547	11.13 ms	9.56 ms	1.67 min	1.35 min
128,197	16.67 ms	14.17 ms	2.54 min	2.09 min
160,545	25.04 ms	21.01 ms	3.43 min	2.80 min
192,816	33.30 ms	27.67 ms	4.58 min	3.73 min
225,262	43.06 ms	35.8 ms	5.61 min	4.44 min
257,694	52.12 ms	44.51 ms	6.90 min	5.36 min

Table 3.2: Computation time per iteration step (in miliseconds) and computation time to form a cloud (in minutes) in function of its number of particles.

3.7.5.2 Several Clouds

The second tests were focused on the generation of several clouds over a terrain. Taking into consideration that we are not using any kind of CPU/GPU core-based acceleration, the results shown in Fig. 3.20 indicate that the generation of cloud scenes with about 50,000 particles (see Fig. 3.19(a)) takes less than a minute. We also created simulations for cloud scenes with 1,000,000 particles, like those shown in Figs. 3.19(b)-(c).

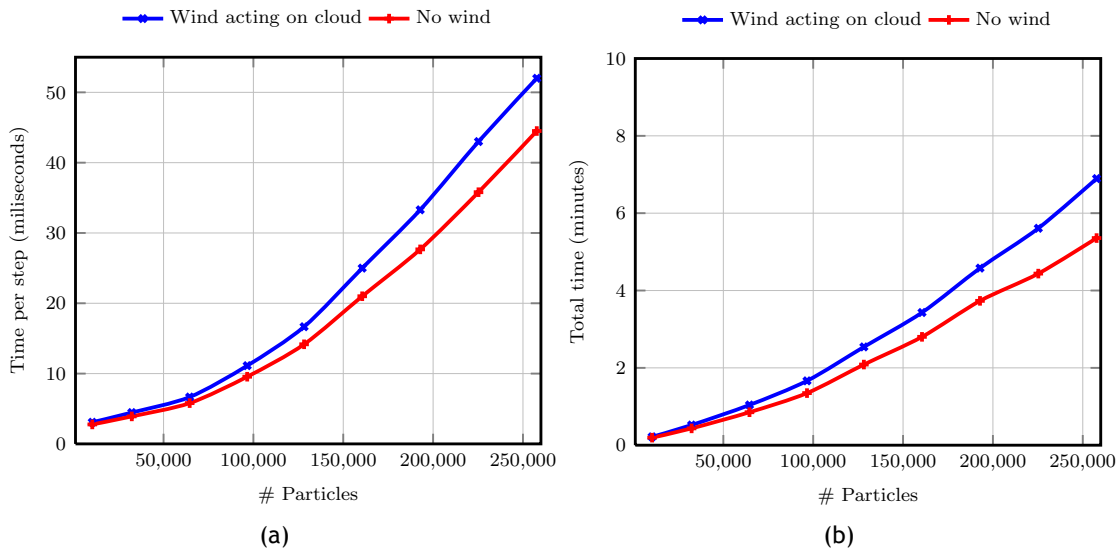


Figure 3.20: 3D cloud simulator times: (a) time spent in each simulation time step; (b) total time spent to generate a cloud with an increasing number of particles.

3.8 Limitations and Future Work

Our work on cumulus clouds using the SkewT/LogP thermodynamic model can be further developed in the following directions:

- *Cloud dissipation*. Cloud dissipation can also be implemented using SkewT/LogP diagrams. The dissipation of particles in the cloud, as well as dynamic formation of new clouds, can be determined by the analysis of the variation of the mixing ratio over time across several sounding data, which reflects changes in the ambient temperature resulting from convection.
- *Wind modelling*. We used a simplified form of a wind model, based on sounding data. Adding vortical structures causing plumes and roiling motion likely endows more realism to clouds. Unfortunately, SkewT/LogP diagrams do not provide enough information to determine the influence of this factor in cloud evolution. Nevertheless, SkewT/LogP diagrams allow us to foresee a storm.
- *Other types of clouds*. We have shown that convective cumulus clouds can be formed on computer using SkewT/LogP thermodynamic diagrams. In addition,

the next chapter describes how to synthetically form orographic clouds using SkewT/LogP thermodynamic diagrams once again. In the case of orographic clouds, we have to take into account other factors such as terrain proximity, windward winds (which force the air mass to rise) and leeward winds (which force the air mass to descend). In the near future, we also aim at simulating other types of clouds such as, for example, cirrus and stratus clouds.

- *Artistic cloud sketching and modelling*. This has to do with cloud shape control, which is a difficult task using physically-based methods. Although our system offers some degree of control to the simulation of cloud shapes, some work needs to be done to provide artists a more intuitive control. The application of the technique introduced by Dobashi et al. [DKNY08] can be a good starting point to first allow artists to define a cloud silhouette and then apply the cloud simulation model presented in this chapter to "fill" the cloud. In terms of physics, this is equivalent to consider a cloud as a fluid with boundaries. This is in line with the suggestion above of using real functions to describe particle release patterns over the interval of convective temperatures for cloud particles.

It is clear that we may also improve the performance of the SkewT/LogP algorithm for cumulus clouds using some kind of parallel processing, namely multi-threading via CPU or GPU.

3.9 Concluding Remarks

In this chapter, we have demonstrated the use of SkewT/LogP diagrams for realistic simulation and rendering of convective clouds in real-time. The SkewT/LogP diagrams are the core of our cloud simulator. These diagrams allow us to geometrically determine the parameters required to solve equation Eq. (3.8), as needed to simulate 3D clouds in real-time.

Therefore, our physically-based approach does not involve the massive computations necessary to solve differential equations of motion of clouds in the atmosphere. This means that our system can be easily incorporated in flight simulators or games as a plugin replacement for methods that use manual modelling of cloud meshes. Moreover, our method can be extended to other types of clouds such as orographic clouds, stratus, cirrus, and fog.

Submitted Paper

The simulation algorithm described in this chapter concerns the convective cloud formation using SkewT/LogP diagrams has originated a paper provisionally accepted for publication, as follows:

> **Rui P. Duarte, José F.M. Morgado and Abel J.P. Gomes: Real-Time Simulation of Cumulus Clouds through SkewT/LogP Diagrams, *Computer Graphics Forum* (submitted after major revisions).**

Chapter 4

Orographic Clouds

During the past three decades much work has been carried out in the formation of convective clouds, due to their multiple applications in film animations, computer games and flight simulators. The previous chapter approached a formation and simulation method for cumulus clouds. Instead, in this chapter, we describe a formation and simulation method for orographic clouds within synthetic 3D scenes. Orographic clouds are those that form under the influence of lofty terrains, i.e., hills and mountains. Furthermore, as far as we know, this is the first attempt to simulate orographic clouds in computer graphics and related disciplines. As in the previous chapter, we shall use sounding data retrieved from weather agencies together with SkewT/LogP diagrams to explicitly solve the motion equation that regulates the behaviour of clouds in the atmosphere. Once again, each cloud is here considered as a particle system.

4.1 Introduction

The simulation of natural scenery containing clouds has always been a hot research topic in computer graphics. Chapter 3 describes a simulation method for cumulus clouds, which are convective, while the current chapter focuses on orographic clouds, i.e., clouds that form under the influence of terrains. To our best knowledge, this is the first attempt to simulate the formation of orographic clouds in computer graphics. As in Chapter 3, our orographic cloud formation method also *explicitly* solves the motion equations using sounding data obtained from weather agencies together with SkewT/LogP diagrams. The difference remains in the fact that one calculates a different set of parameters to solve the motion equation. Additionally, we have to determine the influence of the terrain in the cloud formation process, which is determined using a geometric procedure that accounts for the particle position, horizontal force acting on the particle, terrain geometry, and type of terrain.

As illustrated in Figure 4.1, the major contributions of the work carried out in this chapter are as follows:

- *Use of real sounding data to solve physically-based equations.* To overcome the drawbacks of the two main categories of cloud simulation methods (physically-based and procedural), we explicitly solve the motion equations with sounding data obtained from weather agencies. These atmospheric data are provided in a tabular format, which were incorporated into our two-dimensional SkewT/LogP

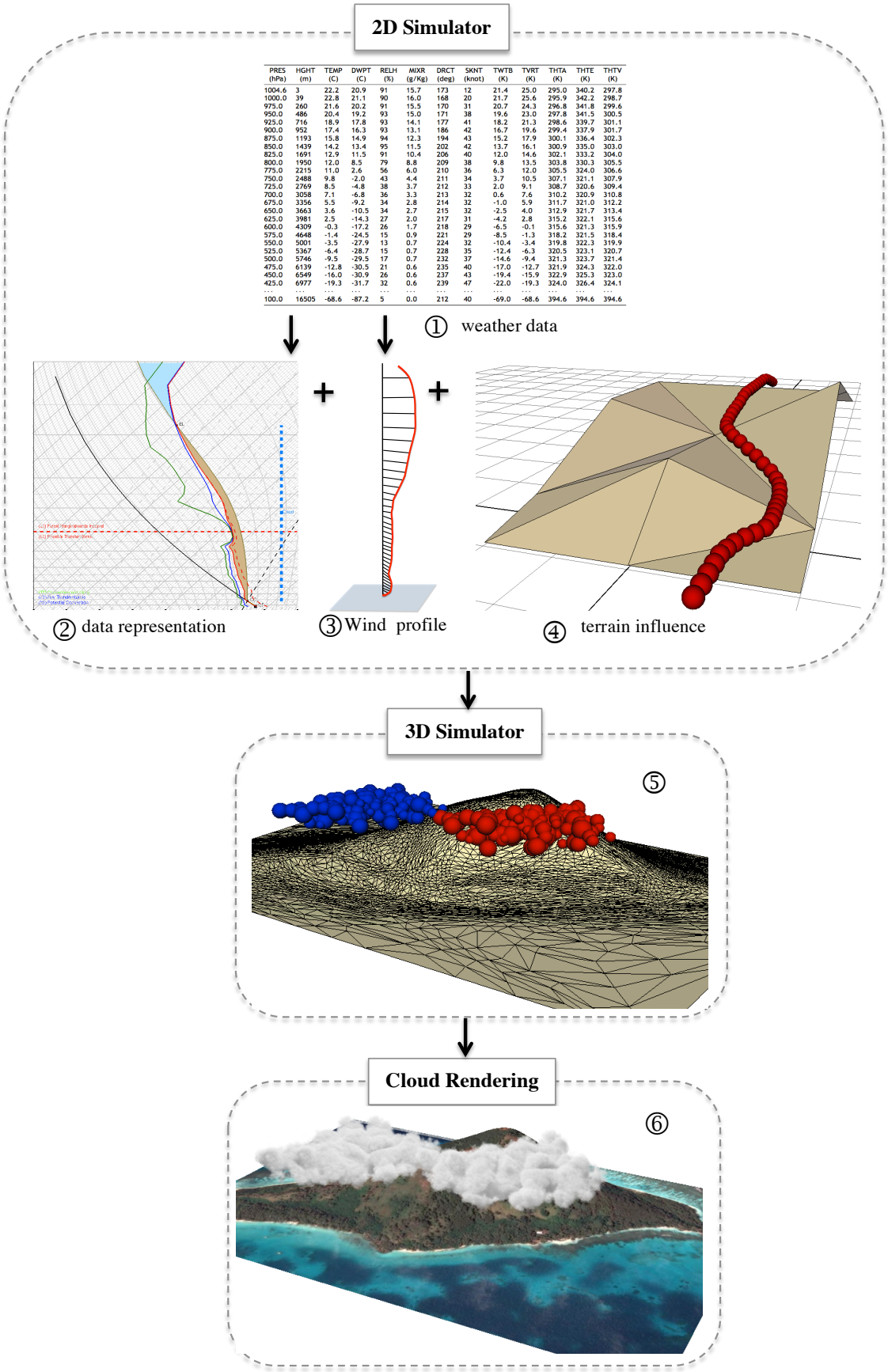


Figure 4.1: Overview of our system: ① input atmospheric data collected by weather agencies; ② sounding data feed the SkewT/LogP diagram to estimate parameters associated to cloud formation; ③ wind profile generated from sounding data; ④ interaction with terrain is determined; ⑤ particles of air parcels rise in the atmosphere and ⑥ are rendered into clouds.

system to determine all the parameters necessary to simulate the forced lift of clouds caused by lofty terrains.

- *Simulation of orographic clouds.* As far as we know, this is the first method that simulates cloud dynamics over lofty terrains (i.e., hills and mountains). This method follows a two level approach: first, clouds are simulated according to the parcel theory, and then we have to take into account their interaction with a terrain in order to generate visually appealing clouds.

The remainder of this chapter is organised as follows. Section 4.2 briefly reviews important works in cloud simulation and rendering. Section 4.3 approaches the physics of clouds, with a focus on the dynamics of orographic clouds. Section 4.4 details the thermodynamic diagrams used to advect an air parcel. Section 4.5 describes the formation of clouds and their interactions with terrain. Section 4.6 puts forward relevant results produced by our cloud simulator. Section 4.7 approaches the limitations of the method. Section 4.8 concludes this chapter with some hints to future work.

4.2 A Brief Overview of Related Works

Most methods available in the literature focus on cumulus clouds with some exceptions on high level clouds such as, for example, stratus and cirrus [KPH+03, BNL06, US10]. This is related to the fact that cumulus clouds are the most common in daily life, with the additional advantage of exhibiting a volumetric shape with flat bases and wispy edges or boundaries, what make them geometrically adequate in simulation and rendering. This assumption let aside other types of clouds, namely orographic clouds. In this case, with the exception of Pixar's short film *Lava* [MLH+15] recently released, no work has been presented in simulating cloud evolution over mountains. Although they present realistic simulations of cloud evolution in their film, no information was made public of how clouds are formed and interact with mountains.

To determine the interaction of a cloud with a mountain, we have to deal with collision detection between them over time. The problem of collision detection has been extensively studied in the literature, in particular those studies concerning collisions between deformable models [VT00, MKE03, BWK03, CTM08], surgical simulations [LCN99, GD04, RGF+04], fluid-solid interactions [SAC+99, MST+04, PPLT09] and robotics [RH02, MM06, DBS+07, SH08]. Recent reviews on different algorithms are approached in [Eri04, TKH+05, JTT01]. Jiménez [JTT01] classified collision detection in four categories: spacio-temporal intersection, swept volume interference, multiple interference detection and trajectory parametrization. In our work we use the trajectory parametrization once we determine the collision instant analytically by explicitly expressing the trajectory of the object as a function of time. In our method, we use a collision prediction method between cloud particles and triangle meshes featuring terrains.

In this context, Stora et al. [SAC+99] proposed a simplified model of lava flow where the terrain is defined by a digital elevation model and collision between particles, while the terrain is reduced to the computation of the terrain altitude at the projection of the particle onto the horizontal plane. In Müller et al. [MST+04], fluid particles interact with deformable objects represented by triangle meshes. In their work, fake particles are placed temporarily on triangles nearby each cloud particle in order to reduce the triangle-particle interaction problem to a particle-particle interaction problem. To avoid the use of these fake particles, Harada et al. [HKK07] developed a method to simulate particle-based flows in complex meshes by using repulsive forces, which are computed by wall weight distance functions between particles and mesh polygons. Paiva et al. [PPLT09] compute the collision between particles and triangles to represent a viscoelastic fluid/solid simulation. The intersection between particles and triangles is made through a simplified version of the algorithm proposed by Karabassi et al. [KPTB99]. To take into account the particle's volume, each segment of the trajectory is modelled as a cylinder with spherical caps, which is the trace of a moving sphere. Collisions are thus handled as cylinder-with-spherical-caps/triangle intersections.

In our method, we use a similar approach. However, we consider that the particle trajectory is defined by an external force (wind) that determines the behaviour of each air particle. Moreover, we account for the terrain influence on the particle behaviour since this terrain interaction starts when the particle is not near the mountain. Summing up, in this chapter, we extend the cloud simulation method described in Chapter 3 to orographic clouds, in the sense that a distinct set of vertical motion parameters is calculated, being also considered the interactions between cloud particles and mountains.

4.3 Cloud Physics

A wide variety of clouds formations is visible in the sky every day. These formations differ at a point that it seems almost impossible to categorise them. However, the World Meteorological Organization (WMO) updates and classifies all the different f cloud ormations that can be observed in Earth's sky. This serves as an international guideline for cloud classification [Mas57]. In this chapter, we will focus on orographic clouds, which usually result from the motion of forced air against lofty terrains, as shown in Figure 4.2.

4.3.1 Atmospheric Dynamics

In Chapter 3, we realised that the process of air mass convection is governed by the laws of thermodynamics. These laws dictate how the convective clouds move upwards in the atmosphere. In regard to orographic clouds, the motion equations are the same

as those presented in Chapter 3. For a detailed explanation of these equations, the reader is referred to Section 3.3.2 and Section B.3 (see Appendix B).

4.3.2 Orographic Clouds

Conceptually, when clouds that form over a lofty terrain, dry air is forced to move from a low elevation to a high elevation driven by the synoptic-scale pressure field by forced convection. As it ascends, it cools dry-adiabatically while saturation is not reached, a point at which condensation occurs. At this point, called the lifting condensation level (LCL), the air parcel has negative buoyancy, but it is forced to rise moist-adiabatically while the air does not attain the level of free convection (LFC). Here, the air parcel is no longer forced to rise, and so it starts rising on its own along its moist adiabat due to positive buoyancy. Finally, it reaches the equilibrium level (EL), which, in this case, is located around the crest of the mountain range. This process is predicted by the parcel theory [Nor38]. According to the science of meteorology, clouds that form under these circumstances have a high likelihood of precipitation occurrence.

As the air descends on the leeward side of the mountain, it is heated dry adiabatically; as a consequence, the air becomes very dry and reaches temperatures that —at an equal elevation— are greater than the original temperature on the windward side. Therefore, the air on the leeward side typically is very dry and clear; as a consequence, the leeward side is seen as a region of high visibility, also called foehn window. Over the crest, the wall of clouds as observed from the leeward side slopes is called foehn wall, as shown in Figure 4.2.

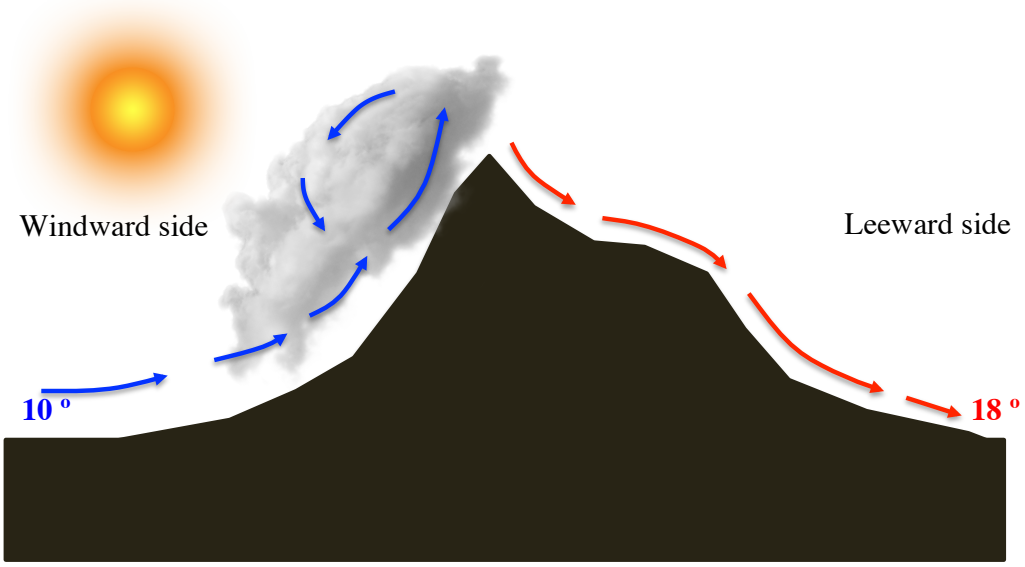


Figure 4.2: Foehn diagram.

Foehn is a generic term for a downslope wind that is strong, warm and very dry. While the term originated in the Alpine area, these winds occur over the world and can also be called *Chinook* (North America) or *Helm wind* (United Kingdom).

Air that impinges on a mountain can either rise over it (unblocked flow) and/or detour it (blocked flow). The air flow response largely depends on the Froude number (F_r), which is given by:

$$F_r = \frac{U}{NH}, \tag{4.1}$$

where U is the upstream wind speed, H is the mountain height and N is the Brunt-Väisälä frequency, or buoyancy frequency, given by:

$$\begin{aligned} N^2 &\approx \frac{g}{\rho_0}\frac{d\overline{\rho}}{dz} \\ &\approx \frac{g}{\theta_0}\frac{d\overline{\theta}}{dz} \end{aligned} \tag{4.2}$$

where $\overline{\rho} = \overline{\rho}(z)$ and $\overline{\theta} = \overline{\theta}(z)$ are the vertical density and potential temperature profiles of the flow, while ρ_0 and θ_0 refer to the density and potential temperature values at ground level, i.e., P_0. Note that F_r represents the ratio of the kinetic energy of the impinging flow to the potential energy required to ascend towards the mountain crest. Since density decreases with height, N^2 is usually positive and the atmosphere is stable. This means that when particles are displaced vertically, they accelerate back towards their initial position. When stable air is forced to ascend a mountain, it becomes negatively buoyant and is pulled down.

When $F_r > 1$, the impinging flow possesses enough momentum to overcome the negative buoyancy that it acquires over the upwind slope, allowing it to complete its ascent towards the mountain crest. On the other hand, if $F_r < 1$, the flow is too weak to overcome the effects of negative buoyancy and detours around the mountain. In this blocked regime, low level winds tend to lie parallel to the height of terrain contours, rather than parallel to them, which causes the flow to move around the mountain.

4.3.3 Wind

In an orographic system, wind is of major importance since the motion of particles greatly depends on the wind profile, more specifically between ground level (where particles are generated) and the level where they rise freely in the atmosphere. The dependence of the wind profile is due to the fact that particles that rise between these levels have negative energy, which would make them sink to the initial position. Considering the vertical wind shear, particles are forced to rise into the atmosphere to the point where they have positive energy. As in Chapter 3, our wind profile is extracted from the sounding data (see columns 6 and 7 in Table 4.1). The wind vector is applied to the particles as shown in Algorithm 7 and Algorithm 8.

4.4 Basic Idea

Cloud formation is due to differences in the temperatures of the rising air parcel and the atmosphere. As we have seen in the previous sections, these differences are well established by using temperatures, specially, potential temperature. With this input, one may determine the cloud development in a more accurate manner, that is, its formation and evolution.

So, the idea was to use SkewT/LogP diagrams introduced in Chapter 3 to extend its applicability to other categories of clouds, in particular orographic clouds. The nature of the vertical motion in an orographic system is different from the one in a convective system. As referred in Section 4.3.2, different levels have to be determined in the atmosphere to feed and to explicitly solve the upwards motion equation (3.8) in order to realistically simulate and render the formation of clouds.

Moreover, in an orographic system, wind plays a major role in the vertical motion of particles while the LFC is not reached, as presented in Section 4.3.3. Also, sounding data obtained for an orographic region is represented in a SkewT/LogP diagram to feed the motion equation. The sounding curves C_a and C_{dew} concerning the ambient temperature T_a and dew-point temperature T_d, respectively, are specifically generated from the data listed in the 3rd and 4th columns shown in Table 4.1.

Table 4.1: An example of an atmospheric sounding in tabular format, obtained from http://www.twisterdata.com/ related to an orographic region.

PRES (hPa)	HGHT (m)	TEMP (°C)	DWPT (°C)	RELH (%)	MIXR (g/kg)	DRCT (deg)	SKNT (knot)	TWTB (°C)	TVRT (°C)	THTA (K)	THTE (K)	THTV (K)
986.8	409	21.2	15.8	71	11.5	330	5	17.8	23.3	295.5	328.9	297.5
975.0	349	21.6	15.4	68	11.4	342	7	17.7	23.6	296.9	330.2	298.9
950.0	574	20.4	13.6	65	10.4	340	7	16.1	22.2	297.8	328.2	299.7
925.0	804	19.5	11.0	58	9.0	347	7	14.3	21.1	299.2	325.8	300.8
900.0	1039	19.1	8.4	50	7.8	8	6	12.7	20.5	301.2	324.5	302.6
...	...	...	...	...	...	...	...	...	...	...	...	...
100.0	16512	-74.0	-84.0	20	0.0	242	59	-74.3	-74.0	384.2	384.2	384.2

4.5 Orographic Cloud Simulation

In a particle system, each particle crosses three consecutive stages: birth, life and death. In this chapter, we refer to the two first stages, since we aim at cloud formation, not cloud dissipation. Cloud formation has to do with the upwards motion of air parcels, which at some point turn into clouds. Therefore, cloud formation comprises the first stage (birth) and part of the second (evolution). On the other hand, cloud dissipation refers to the second part of life and death. Moreover, since terrain elevation plays an important role in cloud evolution, due to its interaction with the cloud formation process, we present a method that accounts for this interaction during the air parcel convection process, as presented in Figure 4.3.

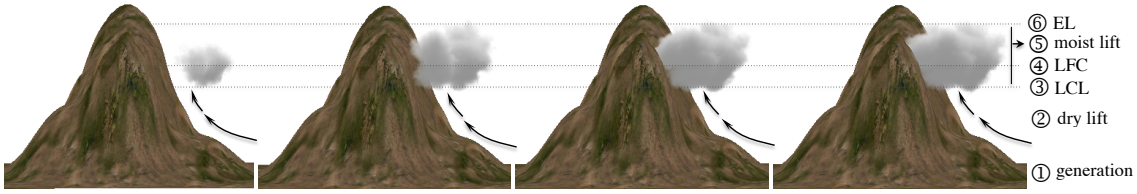


Figure 4.3: Overview of the cloud formation algorithm for a cloud with 5,000 particles: ① particles at ground level are forced to lift due to wind; ② these particles rise in the atmosphere at the dry adiabatic lapse rate. ③ when this parcel rises, it reaches the LCL, a level at which the condensation occurs and the moist lift starts; ④ ⑤ when the parcel attains the LFC, particles are no longer forced to lift by the effect of the wind, so that the moist lift has the forced stage from LCL to LFC and the free stage from LFC to EL; ⑥ particles rise freely until EL is reached.

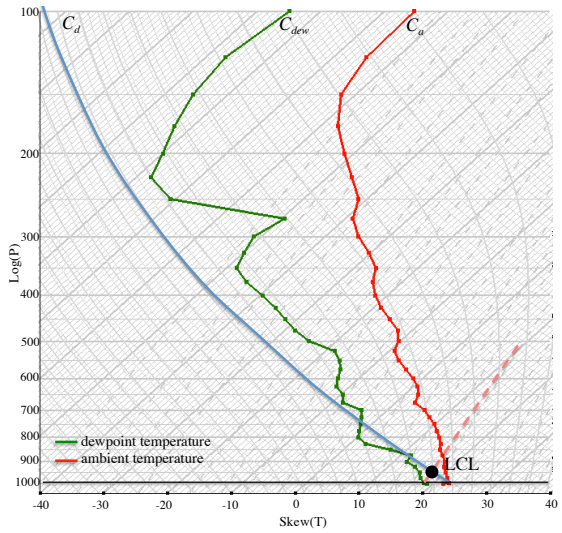
4.5.1 Generation

At ground level, particles of an air parcel Π are randomly generated around one or more emitters located on the ground, i.e., at different positions (x, y, z_0), where z_0 corresponds to the altitude relative to the ground pressure P_0. For each particle π_i, a reference ambient temperature value is generated in the interval $[T_0 - \Delta t, T_0 + \Delta t]$, where (T_0, P_0) refers to the first (or ground) point of the curve C_a, and Δt is a predefined temperature step. In an orographic uplift, these particles are forced to climb the mountain. Therefore, cloud birth terminates with the release of parcel particles at P_0.

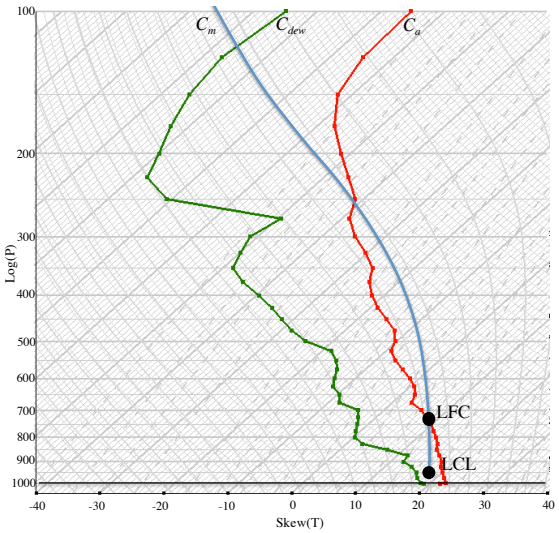
4.5.2 Estimation of Particle parameters

Using a SkewT/LogP diagram, and for each particle generated on the ground, we are able to estimate each of the standard pressure levels that characterise its evolution in the atmosphere: LCL, LFC and EL. In addition, we need to determine the dry and moist adiabatic curves, C_d and C_m, of each particle that ascends in the atmosphere.

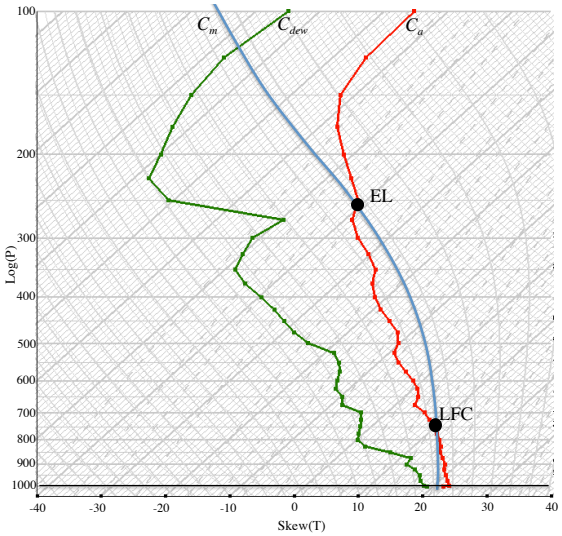
The lifted condensation level (LCL): This is the point $(T_{\mathsf{LCL}_i}, P_{\mathsf{LCL}_i})$ that corresponds to the height at which a parcel of air becomes saturated when is lifted dry adiabatically. It also represents the height of the cloud base. This means that a cloud becomes visible above this level. In a SkewT/LogP diagram, the process is illustrated in Figure 4.4(a) and presented in Algorithm 4. To calculate the LCL point for each particle π_i of a given parcel Π, it is required to determine the intersection between the dry adiabatic curve that passes in the ambient temperature C_a at ground level and the piecewise linear curve l that gives the mixing ratio (6th column in Table 4.1). Thus, we first calculate the potential temperature $\theta(T_{\mathsf{LCL}_i}, P_{\mathsf{LCL}_i})$ at the ambient temperature curve C_a. This is equivalent to determine the dry adiabat, C_d, that goes through this vertex of C_a, for each particle. This way, we end up having a sequence $\theta_0, ..., \theta_n$ of dry adiabatics for the n particles of the air parcel Π. The method used to determine the dry adiabat is similar to Algorithm 2 (presented in Chapter 3, where the CCL is determined for a convective system). In this case, instead of determining the C_d to obtain the CCL, it is determined to obtain the LCL, according to Algorithm 4.



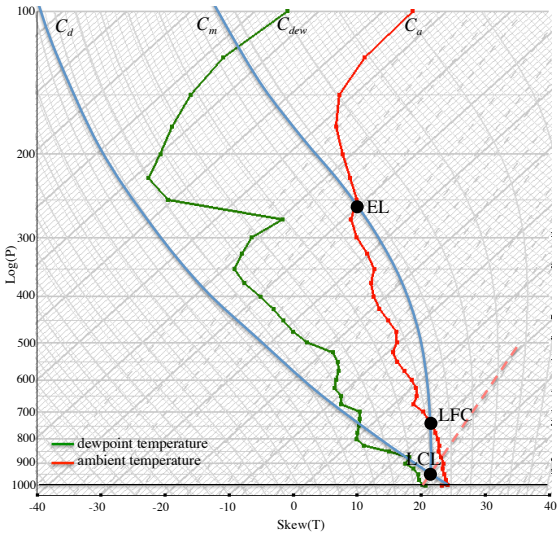
(a) Geometric procedure to estimate the LCL.



(b) Geometric procedure to estimate the LFC.



(c) Geometric procedure to estimate the EL.



(d) All parameters together.

Figure 4.4: SkewT/LogP diagram representation and estimation of several parameters required in orographic cloud formation. The vertical scale is LogP, increasing downwards, and labeled values are given in terms of pressure P: (a) the procedure to estimate the LCL, according to Algorithm 4; (b) finding the LFC, according to Algorithm 5; (c) EL estimation using Algorithm 6; and (d) all three parameters together in a SkewT/LogP.

Considering that the temperature of each particle π_i coincides with the potential temperature of its dry adiabat, we have then to determine the interval $[\boldsymbol{\theta}_i, \boldsymbol{\theta}_{i+1}]$ where T_i fits in. Following a similar approach used to obtain the CCL, after identifiyng the segment $(i, i+1)$ of C_d that crosses the mixing ratio curve of the particle π_i, l, we use a zero finder (from numerical analysis) to determine the LCL point $(T_{\mathsf{LCL}_i}, P_{\mathsf{LCL}_i})$ such that $\theta_{\pi_i}(T_{\mathsf{LCL}_i}, P_{\mathsf{LCL}_i}) = T_i$ (cf. Equation (3.7)). At this point, particles π_i are already assigned a dry adiabatic C_d and a lifted condensation level point LCL_i.

ALGORITHM 4: Computation of the LCL point $(T_{\mathsf{LCL}}, P_{\mathsf{LCL}})$ for a particle π_i

begin

- $(T_{\mathsf{dew}}, P_0) \leftarrow$ dewpoint on the ground ◁ Table 4.1
- $l \leftarrow$ mixing ratio line through (T_{dew}, P_0) ◁ Equation 3.13
- $(T_i, P_0) \leftarrow$ ambient temperature of the particle $\pi_i$on the ground ◁ Table 4.1
- $C_d \leftarrow$ dry adiabat through (T_i, P_0) ◁ Equation 3.14
- $(T_{\mathsf{LCL}}, P_{\mathsf{LCL}}) \leftarrow l \cap C_d$ ◁ intersection between l and C_d

Level of free convection (LFC): Above the LCL, particles continue to be forced to lift while they do not reach their LFC. This level is the height at which a air parcel of air, lifted dry adiabatically until saturation occurs, becomes warmer (less dense and therefore positively buoyant) than the surrounding air and rises freely after this point. To determine this parameter in a SkewT/LogP diagram, the process is described in Algorithm 5 and is illustrated in Figure 4.4(b).

For a particle π_i, LFC_i results from the intersection between the moist adiabatic curve, C_m, passing through the LCL_i and the piecewise linear curve C_a interpolating the sounding ambient temperatures T_a (3rd column of Table 4.1). Given that the LCL_i point is known, we interpolate the moist adiabatic curve that passes through this point using a zero finder (as for the dry adiabat). Next, we evaluate each segment $(i, i+1)$ of C_a in order to determine the segment that intersects C_m. If such intersection occurs, the LFC_i point is calculated. With this procedure, two more parameters are assigned to the particle π_i: the LFC_i, and the moist adiabatic curve related to the moist temperature profile of the particle, C_m.

ALGORITHM 5: Computation of the LFC point $(T_{\mathsf{LFC}}, P_{\mathsf{LFC}})$ for a particle π_i

begin

- $(T_{\mathsf{LCL}}, P_{\mathsf{LCL}}) \leftarrow$ LCL point of particle π_i ◁ Algorithm 4
- $C_m \leftarrow$ moist adiabatic line through $(T_{\mathsf{LCL}}, P_{\mathsf{LCL}})$ ◁ Equation 3.17
- $(T_{\mathsf{LFC}}, P_{\mathsf{LFC}}) \leftarrow C_m \cap C_a$ ◁ intersection between C_m and C_a

Equilibrium Level (EL): The equilibrium level EL_i of a particle is the height (above the LFC) where the temperature of a buoyant rising parcel again becomes equal to the temperature of the environment.

Typically, particles tend to oscillate around this level due to energy gain from the previous step, while equilibrium is reached. The SkewT/LogP calculation of EL is described in Algorithm 6 and illustrated in Figure 4.4(c). From the previous step, the moist adiabatic profile of a particle is determined and given by C_m. The point EL_i, is the first

point, above the LFC$_i$, that results from the intersection between C_m and C_a.

ALGORITHM 6: Computation of the EL point $(T_{\text{EL}}, P_{\text{EL}})$ for a particle π_i

begin
- $(T_{\text{LFC}}, P_{\text{LFC}}) \leftarrow$ LFC point of particle π_i ◁ Algorithm 5
- $C_m \leftarrow$ moist adiabatic line through $(T_{\text{LCL}}, P_{\text{LCL}})$ ◁ Algorithm 5
- $(T_{\text{EL}}, P_{\text{EL}}) \leftarrow C_m \cap C_a$ ◁ intersection between C_m and C_a

This process allows us to determine, for each particle π_i, the levels where they are supposed to condensate (LCL$_i$), rise buoyantly in the atmosphere (LFC$_i$) and reach equilibrium (EL$_i$). Moreover, two curves are assigned to each particle, which represent its temperature profile in the atmosphere: the dry adiabat curve C_d and the moist adiabat curve C_m, as shown in Figure 4.4(d). In the next section, these curves are used to determine the vertical motion of particles in the atmosphere.

4.5.3 Orographic Motion

In an orographic system, an air parcel is forced to rise to the LFC by the action of the wind. Increased vapour content in the atmosphere can result from wind convergence over water or moist ground into areas of upward motion [Pea02]. So, wind convergence forces the air to increase its moisture and air parcels start its motion between the ground level and the LFC. As shown in Figure 4.4, the dry lift has to do with the particle rising from the ground to the LCL. At this level, it is expected that condensation occurs, and when this happens, the air parcel is forced to move moist adiabatically to the LFC, and the moist lift is established.

Between the LFC and the EL particles rise buoyantly and tend to become stationary inside the cloud. Thereof, the entire lift motion (i.e., dry lift and moist lift) of each particle is ruled by Equation (3.8). To solve this equation at a point (T, P) located at altitude z, two values are required, θ_{π_i} and θ_a, as presented in Algorithm 7. The value of potential temperature θ_{π_i} for each particle is constant and known from the previous step. The value of the potential temperature θ_a for the ambient temperature curve C_a, at the same pressure level P is is determined by finding the point (θ_a, P) that stems from the intersection between the isobar P and a segment of curve C_a, which reduces to a line-to-line intersection problem.

4.5.4 Terrain Influence

Orographic influence is a very significant factor in cloud formation and cloud persistence. When a fluid is moving over a terrain lower boundary, the vertical velocity of the fluid will be upward or downward, depending on the horizontal direction of the fluid flow relative to the slope of the bottom topography. In atmospheric dynamics, clouds will form if the air forced over the terrain is sufficiently moist and in the upward direction. Given a terrain, two situations occur: the particle moves around the mountain,

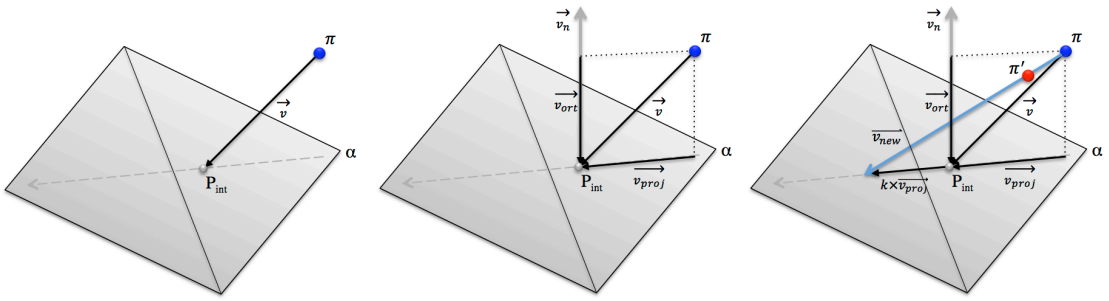
ALGORITHM 7: OrographicMotion($\{\pi_i\}, \Pi, M, C_a, \{T_{c_i}\}$)

Data: $\boldsymbol{\pi}, \boldsymbol{\Pi}$ ◁ particle $\boldsymbol{\pi}$, parcel of particles $\boldsymbol{\Pi}$
Data: $\boldsymbol{C_a}$ ◁ curve that interpolates ambient temperatures T_a
Data: $\boldsymbol{M}$ ◁ terrain mesh
Result: $\boldsymbol{\pi' \leftarrow (\pi'_x, \pi'_y, \pi'_z)}$ ◁ new positions of particles, $\pi \in \Pi$
begin
  $\mathcal{N} \leftarrow$ number of particles in the air parcel Π
  count $\leftarrow 0$ ◁ number of particles at the EL
  while count $< \mathcal{N}$ **do**
    foreach $\pi \in \Pi$ **do**
      $l \leftarrow$ isobar line at pressure P_i
      if $P_i > P_{LCL_i}$ **then**
        $C_d \leftarrow$ dry adiabat curve for π ◁ Equation 3.14
        $(T_A, P_A) \leftarrow (l \cap C_d)$ ◁ get intersection point with C_d
      else if $P_i > P_{LFC_i}$ **then**
        $C_m \leftarrow$ moist adiabat curve for π ◁ Equation 3.17
        $(T_A, P_A) \leftarrow (l \cap C_m)$ ◁ get intersection point with C_m
      $(T_B, P_B) \leftarrow (l \cap C_a)$ ◁ get intersection point with C_a
      $\theta_\pi \leftarrow$ potential temperature at (T_A, P_A) ◁ Equation (3.7)
      $\theta_a \leftarrow$ potential temperature at (T_B, P_B) ◁ Equation (3.7)
      $a \leftarrow 9.81 \cdot (\theta_\pi - \theta_a)/\theta_a$ ◁ Equation (3.8)
      if $a \leqslant 0$ **then**
        $\Delta z \leftarrow (v_i + v_z) \cdot t$ ◁ v_z wind shear
      else
        $v_i \leftarrow v_i + a \cdot t$ ◁ velocity of i-th particle
        $\Delta z \leftarrow (v_i + v_z) \cdot t + 1/2 \cdot a \cdot t^2$ ◁ truncated Taylor series
      $\pi' \leftarrow$ TerrainInfluence$(\pi, \Pi, M, \Delta z)$ ◁ see Algorithm 8
      $P_i \leftarrow (\frac{44331.514 - \pi'_z}{11880.516})^{1/0.1902632}$ ◁ pressure at π'_z
      if $P_i \leq P_{EL_i}$ **then** count = count +1;

following its contour, or goes over the mountain. In Figures 4.5(a)-(c) we present the geometric approach used to determine this interaction, while Figure 4.6 shows how a particle moves over a mesh in order to illustrate the application of the method.

Consider particles π_i of an air parcel Π and a mesh terrain M. The basic idea of our method is to determine the wind direction $\vec{v}$ over the influence of a terrain mesh M. This progression allows us to determine the deviation in the particle trajectory. For example, if the wind vector $\vec{v}$ is near parallel to the mountain, then its projection onto the mountain is big, so that we end up having a significant deviation in the particle position. On the other hand, if the wind vector is nearly perpendicular to the mountain, the projection is small, so the corresponding positional deviation of the particle is small. To determine this impact on the particle behaviour, our terrain detection algorithm is composed of three steps.

In the first step, one determines the faces F of the mesh M that are intersected by a line segment that contains the particle π in the wind direction, as shown in Figure 4.5(a). If more than one face of the mesh is intersected, the method considers the visible face from particle's position, that is, the closest face to the particle π. In the second step, one determines the projection $\overrightarrow{v_{\text{proj}}}$ of $\vec{v}$ onto such face of the terrain mesh, as described in Algorithm 8, and shown in Figure 4.5(b).



(a) Step 1 (b) Step 2 (c) Step 3

Figure 4.5: (a)–(c) Overview of the approach to detect terrain.

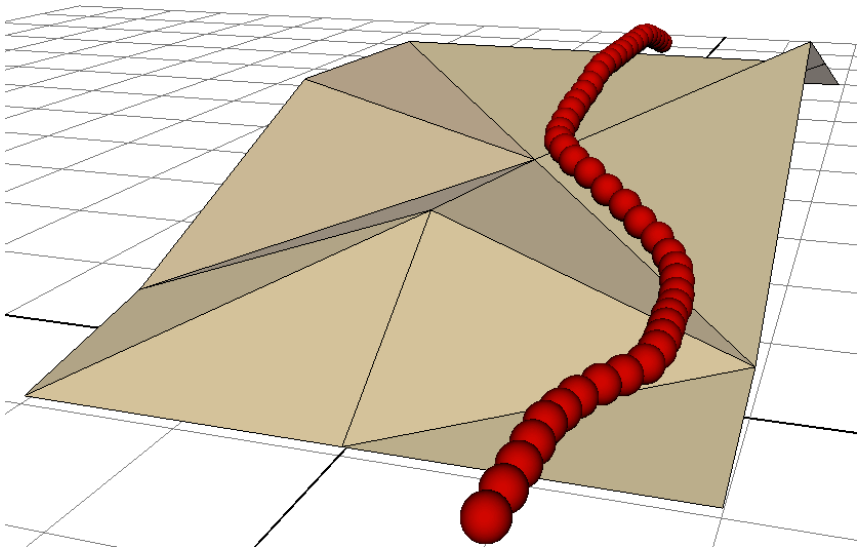


Figure 4.6: Motion of a particle over a triangle mesh.

In other words, $\overrightarrow{v_{\text{proj}}}$ represents the wind contribution parallel to some face of the mountain. Finally, when approaching a mountain, the particle motion depends on two factors: the terrain roughness, $R \in [0, 1]$, and the distance d of the particle to the terrain. Considering this, we have defined an attenuation factor $k = \frac{(1-R)}{d^2}$ for each face of the mountain, which allows to obtain the new wind vector $\overrightarrow{v_{\text{new}}}$ and update the particle position according to such new vector, as shown in Figure 4.5(c).

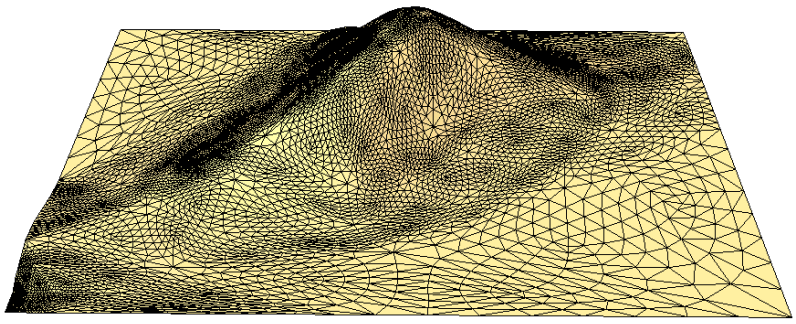
The attenuation factor defines the terrain roughness as follows. It is high ($R \approx 1$) for forrest terrains, while it is small ($R \approx 0$) for clear terrains without natural obstacles or others. On the other hand, if the distance d is too high, k is nearly zero, meaning that there is no terrain influence over the particle motion, so that the new wind vector, $\overrightarrow{v_{\text{new}}}$ is not affected either.

Our simulator uses terrains extracted from google maps. We reduce the complexity of such a terrain mesh by applying the smoothing technique presented in [SG04], i.e., the number of triangles decreases, though the mesh topology is preserved (see Figure 4.7). With this, we are able to speed up the collision detection process, as described in Algorithm 8.

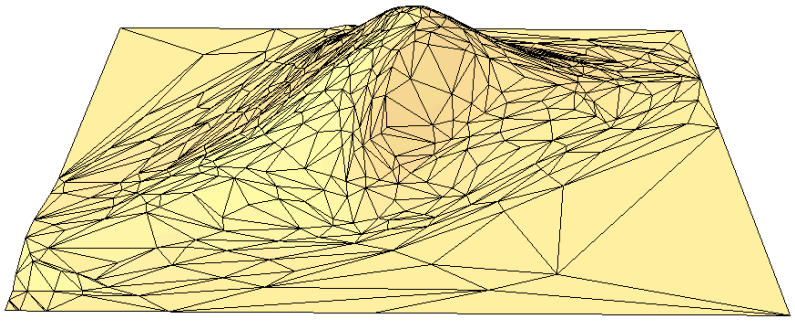
ALGORITHM 8: TerrainInfluence$(\pi, \Pi, M, \Delta z)$

Data: π, Π ◁ particle π, parcel of particles Π
Data: $\vec{v} = (v_x, v_y, v_z)$ ◁ wind vector
Data: F, M ◁ Faces F in Terrain mesh M
Result: $\pi' \leftarrow (\pi'_x, \pi'_y, \pi'_z)$ ◁ new positions of particles, $\pi \in \Pi$

```
begin
    foreach π ∈ Π do
        S ← (π, v⃗)                          ◁ line segment in the wind direction
        j ← 0;
        foreach F ∈ M do
            P_int ← S ∩ F                    ◁ intersect face in the wind direction
            if P_int then
                T[j] ← F                     ◁ store the face in a triangle list
                V⃗_N ← NormalToFace(T[j])
                j ← j + 1;
        foreach T do
            v⃗_ort ← proj_{V⃗_N}(v⃗);
            v⃗_proj ← v⃗ − v⃗_ort;
            v⃗_new ← v⃗ + k × v⃗_proj           ◁ k is an attenuation factor
            π' ← π + v⃗_new + (0, 0, Δz)
```



(a) 14,784 triangles



(b) 869 triangles

Figure 4.7: Reducing the number polygons in a terrain mesh.

This allows for the simulation of the motion of particles in the atmosphere, yet that considering the influence of the terrain geometry. Figure 4.6 illustrates this for the motion of a single particle. This terrain-based approach coupled with the atmospheric model presented in Section 4.5.3 are instrumental in the simulation of the formation of clouds in an orographic system, as illustrated in Figure 4.8.

4.6 Results

Our results were obtained on a desktop computer equipped with and Inter Core i7 with 3.07 GHz, 24 GB of RAM, and a Nvidia Quadro 6000 6GB graphics card, under the control of the Windows operating system (v7) . No parallel computing resources and tools were used in the simulation process, i.e., simulation of the cloud formation was exclusively performed on CPU.

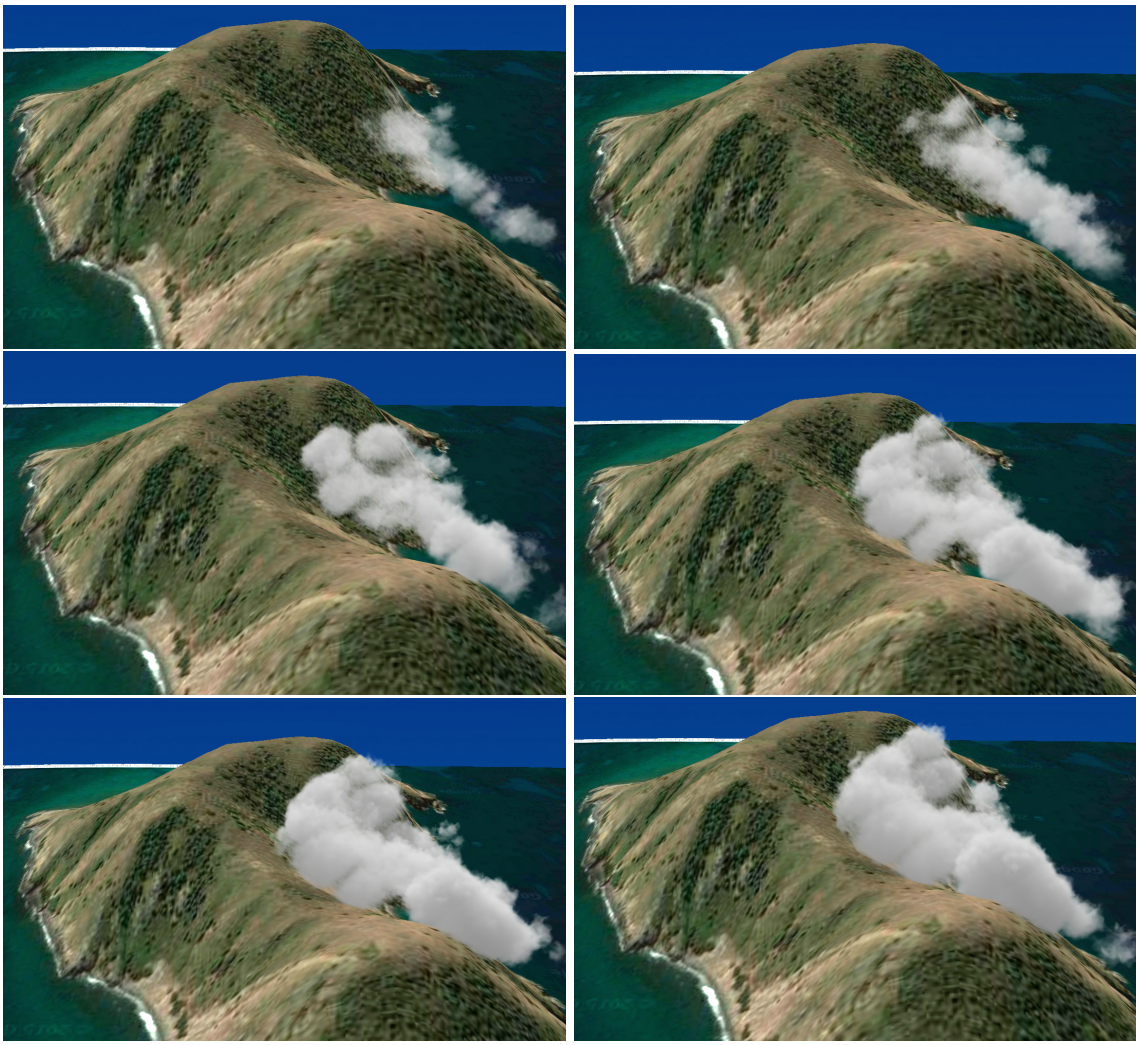
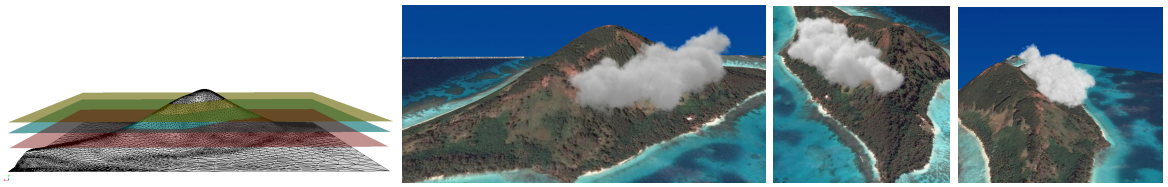
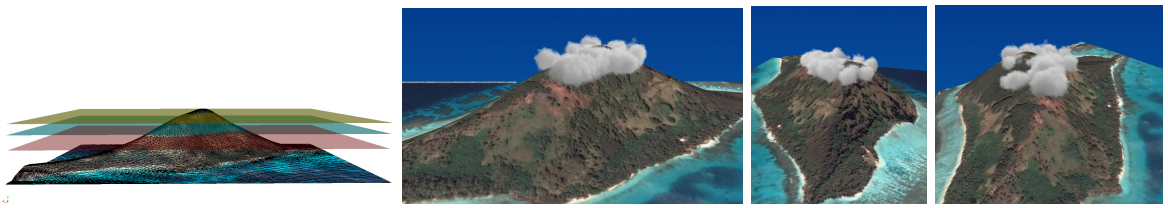


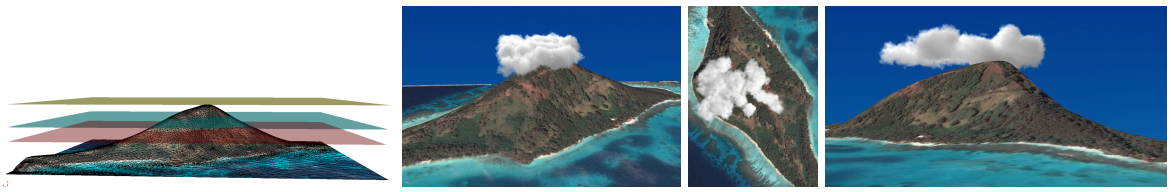
Figure 4.8: Simulation of the formation of a cloud composed with 10,000 particles from ground generation to equilibrium level, with $F_r = 0.9$ and mesh average attenuation factor $R \approx 0.73$.



(a) Cloud formed within the mountain range. $F_r = 0.85$ and mesh average attenuation factor $R \approx 0.92$.



(b) Cloud formed at the mountain top. $F_r = 1.12$ and mesh average attenuation factor $R \approx 0.92$.



(c) Cloud formed above the mountain top. $F_r = 1.24$ and mesh average attenuation factor $R \approx 0.92$.

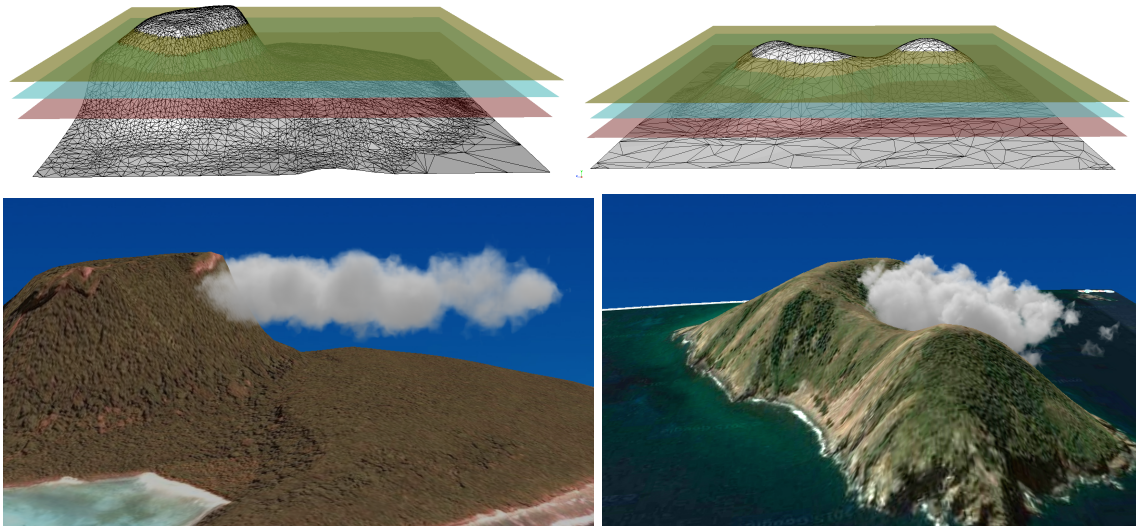
Figure 4.9: Mesh terrain with 14,784 faces and the influence of changing the atmospheric levels in cloud shape. (a) EL is reached bellow the mountain top and clouds do not pass the top of the mountain. (b) EL is reached at the mountain top, and some particles within the cloud reach equilibrium on both sides of the mountain. (c) EL is reached above the mountain top, which implies on clouds forming over mountains.

4.6.1 3D Cloud Simulation and Rendering

A 3D cloud interface was implemented in C++ using the OpenGL/GLUT libraries to simulate cloud formation in the atmosphere over lofty terrains. This simulator uses the data structure defined for the 2D simulator to solve the equation of motion in real-time for particles generated at ground level. As particles evolve in the atmosphere using Algorithm 7, their interaction with terrain is determined using the technique presented in Figures 4.5(a)-(c) for each particle in the system. The realistic rendering of clouds was carried out using Elementacular Maya plugin [Ale15].

To validate our approach, three terrain-dependent scenarios were considered: (i) formation of clouds from ground generation to equilibrium; (ii) interactive change in atmospheric levels; (iii) and terrain topography influence. In the first scenario, the formation of a cloud is simulated from the ground to the level where they reach the equilibrium level, as presented in Figure 4.8. Here, particles evolve in the atmosphere following the dry and moist adiabats determined in the SkewT/LogP thermodynamic diagrams, as explained and illustrated in the previous sections.

The second scenario involved to interactively change the standard pressure levels in the 2D interface, at which clouds form and become stationary, though considering the terrain elevation constraint, as presented in Figure 4.9. Here, three situations were simulated. The first, presented in Figure 4.9(a) refers to the generation of a cloud



(a) (b)

Figure 4.10: Clouds formed under different terrains but with similar temperature profiles: (a) terrain composed of 21,353 triangles with a single; (b) terrain composed of 7,830 triangles with two peaks.

where the altitude of EL does not surpass the mountain peak, since the particles reach equilibrium before reaching the mountain top. Here, all the condensation occurs in one side of the mountain and these clouds become stationary within the mountain range. They are typically referred to foehn walls. The second, presented in Figure 4.9(b), refers to the generation of a cloud where the EL altitude was set to be very near the mountain top. In this case, some particles are forced to move to the other side of the mountain, which results in clouds forming around the mountain peak, also referred to as cap clouds. The third, presented in Figure 4.9(c), has to do with the formation of clouds above the mountain peak. In these circumstances, clouds distribute themselves about the mountain peak since EL is reached at an higher level than the terrain elevation. This shows that by using different atmospheric soundings over the same terrain, we can simulate the formation of different types of clouds that exist in Earth's atmosphere since their interaction with the mountain will differ in conformity with the atmospheric levels reached by particles within an air mass.

The third scenario considers the influence of terrain topography on the evolution of clouds in the atmosphere. Typically, our system is able to deal with all types of terrain interactions of rising particles due to forced lift by wind. Figure 4.10 shows an air mass moving against a mountain in a scenario similar to the one presented in Figure 4.9(a). However, here, since the mountain peak width is smaller and the EL is reached below the mountain top, the cloud is forced to move around the mountain, instead of stabilising within the mountain range.

Combining these scenarios, we have generated two clouds based on two emitters. One cloud is supposed to reach the mountain and not pass its top, while the other cloud is generated in order to reach the mountain at a different location, where particles within the cloud are forced to move around the mountain. This results in two clouds

with different behaviours within the same mountain range, as shown in Figure 4.11.

4.6.2 Performance

We have carried out performance tests of our algorithm in simulating and rendering clouds that form over terrains, as presented in Table 4.2. The testing results show that by increasing the number of mesh triangles of a given terrain, the time required to simulate each step also increases. This is explained by the fact that the collision detection algorithm takes longer to process more triangles per particle belonging to the air parcel. By increasing the number of particles, while preserving the number of terrain mesh triangles, the cost in each simulation step increases, but not in the magnitude of increasing the detail in the terrain mesh. Thus, increasing the terrain detail will increase the complexity of the overall simulation in a more noticeable manner.

Table 4.2: Average computation time (in seconds) per iteration step of cloud scenes with terrain interaction using our method.

#particles	#triangles	average time (s)
5000	800	0.42
10000	363	0.28
10000	235	0.19
15000	413	0.44
25000	802	0.79

4.7 Limitations and Future Work

At this point, we are able to generate several types of orographic clouds under different environment conditions and different terrain geometries. One major bottleneck during the simulation process of cloud evolution has to do with the need to verify all the triangles in the terrain geometry to determine wether a particle intersects the mountain or not. This leads to significant computational costs, in particular when the detail of the terrain mesh increases. When the terrain is not taken into account, the average simulation time step takes 0.00308 seconds for a cloud of 10,000 particles. But, when the terrain is involved in the simulation, the time step is about 0.28 seconds on average for a mesh with 363 triangles, as shown in Table 4.2. Nevertheless, our method is suitable to generate orographic clouds for terrains consisting of a low number of polygons, typically used in games or flight simulators. In the future, we intend to reduce the complexity of the terrain interaction algorithm by determining the neighbour triangles of a triangle that is likely to be intersected by a particle. Furthermore, we intend to parallelise the technique in order to be able to generate scenes with a massive number of particles at interactive rates.

Figure 4.11: Different views of two clouds generated using two emitters on the ground, with each emitter placed on a distinct side of the mountain in relation to its crest, using an average attenuation factor $R \approx 0.92$ and $F_r = 0.91$.

4.8 Concluding Remarks

In this chapter, we have demonstrated the use of SkewT/LogP diagrams for realistic 3D simulation and rendering of orographic clouds. Additionally, these orographic clouds take into consideration the influence of terrain in the evolution of a cloud near to a mountain. Our technique adapts to distinct terrain to realistically generate several types of mountainous clouds. In fact, regarding the control of the cloud dynamics, it cannot be decoupled from the terrain geometry, since clouds will form due to the presence of a mountain. However, the user can control the location where clouds will form, by changing the place where the air mass is generated and adjusting the wind profile. To control the atmosphere profile, the user can adjust several parameters related to cloud formation by dragging the reference levels of LCL, LFC and EL. This implies adjusting the cloud base an cloud top. Moreover, the user is able to interactively adjust the wind profile by dragging the ambient temperature curves obtained from sounding data directly in the 2D SkewT/LogP interface.

Submitted Paper

The algorithm for the simulation of orographic cloud formation originated the following paper:

> **Rui P. Duarte, Francisco Morgado and Abel J.P. Gomes: Orographic Clouds,** ***Computer Graphics Forum*** **(submitted for publication).**

Chapter 5

Conclusions

This thesis builds upon the research work in simulation of cloud formation in 3D scenes, which is very challenging topic in computer graphics. Briefly speaking, one can say that the main contribution of this research work lies in the *explicit* cloud simulation method introduced in the third chapter. This means that the equation that describes the rising cloud motion in the atmosphere is solved explicitly; more specifically, such equation is solved using sounding data as input that varies over time. Additionally, we have shown that this explicit approach is also adequate for other types of clouds (e.g., mountain clouds), so it is opened a new line of research in respect to the simulation of other sorts of clouds, as it is the case of stratus, cumulonimbus, altostratus, cirrus, and so forth.

5.1 Research Context

This thesis has tackled the problem of simulating the generation and formation of clouds in 3D scenes, as needed in computer graphics applications such as movies, games, military simulators, and so forth. Instead of using implicit physically-based methods or procedural methods, this thesis puts forward an *explicit* physically-based method for the first time in the field of computer graphics. As a consequence, it was solved the typical problem of physically-based methods that has to do with the lack of real-time processing of the dynamics of a cloud rising in the atmosphere. This is so because solving Navier-Stokes equations of a fluid like a cloud in the atmosphere is a very time-consuming task.

We overcome this problem by feeding the equation of the rising cloud motion with real atmospheric sounding data acquired by atmospheric balloons. In fact, we were able to simulate the formation of cumulus clouds at interactive rates using physics, but avoiding solving the Navier-Stokes equations, provided that, instead, one uses SkewT/LogP thermodynamic diagrams together with those sounding data. Furthermore, we have extended our method to simulate other types of clouds, as it is the case of orographic clouds. In this context, we have also developed a method that accounts for the forced lifted of air parcels due to the influence of terrain.

5.2 Research Questions, Results, and Contributions

We are now in position to answer the research questions put forward in Chapter 1. Let us then remind them herein in order to be more effective in the corresponding answers:

Is it possible to simulate the formation of physically-based clouds in real-time?

In Chapter 3, we proposed a model based on the parcel theory to simulate the dynamics of formation of cumulus clouds. This model is based on physics but, unlike other physically-based methods, it solves explicitly the equations of cloud motion. To this end, we introduced SkewT/LogP thermodynamic diagrams in the field of computer graphics —which are commonly used in meteorology science—, as an attempt to overcome the computational burden of physically-based methods. The use of these diagrams allows us to represent vertical profiles of the ambient temperature in the atmospher using real sounding data retrieved from weather agencies. Given these atmospheric data, parcel profiles can be determined and used to explicitly solve the motion equation for clouds in real-time with parameters extracted from the thermodynamic diagram. These diagrams are implemented as a 2D interface, from which one feeds the 3D simulator where clouds are synthetically formed, from their inception on the ground until their equilibrium level in the atmosphere. Therefore, this chapter answers the first two questions, namely:

Can real meteorological data be incorporated into a model to simulate the formation of clouds?

As seen in Chapters 3 and 4, such meteorological data can be used to feed atmospheric diagrams, from which we can extract parameters related to pressure and temperature that allow us to explicitly solve the motion equation for clouds in real-time. Furthermore, we proved that feeding skewT/logP diagrams with such data makes it possible to generate two different sorts of clouds, namely convective clouds and orographic clouds.

Is it possible to develop a method that can generate various types of clouds?

In Chapter 4, we showed that the method described in Chapter 3 can be extended to orographic clouds, which are not convective. Indeed, the wind blows making particles of a given air mass to rise from the ground against hills and mountains, forcing the air mass to lift in the atmosphere. These new conditions in the formation of clouds means that other parameters need to be calculated using the skewT/logP diagrams in order to simulate the formation of orographic clouds. Furthermore, once orographic clouds are influenced by the terrain elevation, we have also developed a method that accounts for collision detection between particles within an air mass in motion and a terrain mesh representing a mountain. The combination of these two algorithms makes it possible to simulate the formation and evolution of orographic clouds under diverse scenarios.

How to control the final shape of generated clouds?

The fourth question relates to fluid control. Using SkewT/LogP thermodynamic dia-

grams several aspects related to cloud shape and cloud evolution can be controlled. By changing the temperature of generated particles at ground level, it is possible to control of the height and distribution of clouds, i.e., the system is able to generate flat/higher clouds, turret-shape clouds, and upside down turret-shape clouds; for example, if the user wants to create a double turret-shape cloud, it is enough to define two particle emitters at ground level where particles are generated. If the user requires a large-scale cloud, a larger area for emitted particles on the ground is necessary. For this we need to use an 2D interface slider that allows for enlarging/shrinking the area on the terrain of the 3D scene. The cloud base and top can also be specified by the user. In a convective system, the user can pick up and drag the reference CCL and EL points within the SkewT/LogP diagram in an interactive manner. In an orographic system, the same applies to LCL, LFC and EL. To control the overall dynamics of a cloud system, the user can also pick up and drag the sounding curves. To control the wind effect on the cloud shape, the user can also pick up and drag the wind profile curve in the 2D thermodynamic simulator.

Figure 5.1: The photograph used as a motivation for this thesis (see page 5, Figure 1.1), where a cloud generated from our system (in the left) in integrated to show that the grand objective of this thesis has been achieved: the generation of realistic and visually appealing clouds. Far clouds are a part of the photograph and are not generated by our simulator.

Summing up, and taking into consideration the thesis statement put forward in Chapter 1:

> *Is it physically possible to use real data to simulate the dynamics of cloud formation in real-time on CPU?*

we can say that the research work described in this thesis shows that the answer to this question is positive. This answer is sustained on the principle of *vertical displacement*

of air. This principle led to the algorithms presented in Chapters 3 and 4, which allowed us to simulate the vertical evolution of a cloud in the atmosphere by explicitly solving the motion equation for clouds under different conditions; more specifically, the convective clouds result from solar heating of the ground (cf. Chapter 3), while orographic clouds stem from the forced lift of air masses over hills and mountains (cf. Chapter 4). In both situations, this was accomplished using sounding data made available by weather agencies; these data work as input for SkewT/LogP thermodynamic diagrams, from which one determines several atmospheric parameters to feed the motion equation. Therefore, the grand objective of this thesis has been achieved since we introduced a new approach to the simulation and realistic visualisation of cloud formation in computer graphics that satisfies the aforementioned principle of vertical displacement of air, as illustrated in Figure 5.1.

5.3 Discussion, Limitations and Future Work

We have shown that SkewT/LogP diagrams and real data obtained from atmospheric soundings may work as a basis to form clouds in the atmosphere, which start to be formed by releasing air particles on the ground. In the following, we discuss the limitations of our approach, which also constitute an opportunity for future work.

5.3.1 Wind Field

Sounding data include wind data in the atmosphere, the so-called wind profile. This profile provides us the horizontal wind intensity and direction at successive levels of pressure (or altitude), and inherently the vertical wind shear that is determined by the changes of wind vector across successive levels of pressure. However, this external force acting on a cloud is exerted locally because is given by the rising path of a meteorological balloon in the atmosphere. To solve this problem, one can use other available data that feature wind evolution in a region, instead of a localised profile. Moreover, wind advection could be improved by coupling such data with 3D Perlin noise to simulate turbulence and Rayleigh-Bénard convection in the actual air currents [Get99]. Both approaches would provide more realism to the dynamic flow of synthetic clouds.

5.3.2 Control of the Shape of the Clouds

In physically-based simulations, the evolution of the system and its outputs are entirely controlled by the physics laws of nature. As seen in Chapter 3, a cloud in nature is a fluid with boundaries, but we can control its base, height, and top by changing the temperature curves and the particle temperature ranges on the ground.

In order to have more control on the shape of the clouds, we need to add boundaries

to the clouds. For example, the technique due to Dobashi [DKNY08] is a good starting point to merge the concepts of ”easy to define” cloud shape (i.e., 3D cloud silhouette) and simulation methods that account for some sort of physics. Using cloud silhouettes is equivalent to impose a boundary to the cloud top, so we do not see why this cannot be done in computer graphics, with the advantage of still having cloud dynamics regulated by physics. Another advantage in using cloud silhouettes is that they can be obtained from artist’s sketching, what opens a window to combine cloud physics and art in synthetic outdoor scenes.

5.3.3 Simulation of Cumulonimbus and Layer clouds

Our method is based on the parcel theory. This allowed us to generate cumulus clouds, which are convective clouds. Furthermore, we showed that such theory can be handled in a way to simulate the behaviour of orographic clouds. Based on the conditions that allowed us to form synthetic clouds in 3D scenes, we believe that our method is suitable to generate other types of clouds, namely cumulonimbus and layer clouds, shown in Figure 5.2(a) and Figure 5.2(b). The first are related to thunderstorms clouds that cover up to 24 km at ground level and produce thunder and lighting. They form at altitudes between 200 m to 4,000 m and their peaks can reach to as much as 6,000 m. They are characterised by a flat, anvil-like top caused by wind shear. Often, rising parcels surpass the equilibrium level due to momentum. Their dynamics is based on convective systems and at the mature stage, they can turn into supercells, where a mesocyclone occurs, that is, a persisting rotating updraft, typically established by a vortex. Given the internal dynamics of these clouds, we believe that coupling our convective model with a vortex-based system, we will be able to generate cumulonimbus.

(a) Cumulonimbus clouds. (b) Layers of clouds.

Figure 5.2: (a) A huge thunderstorm supercell photographed from NASA’s DC-8 airborne science laboratory as it flew at an altitude of 12,000 m southwest of Oklahoma City. NASA (b) Cloud layers taken from airplane near Mount Fiji. Jeffrey Friedl

Layers of clouds are clouds that form in different layers of the atmosphere, in the same region and can be picked up from examining the SkewT/LogP diagram. Typically, this type of clouds are present in the atmosphere when temperature and dewpoint just above the boundary layer (layer near the ground affected by diurnal heat, moisture or

momentum transfer to or from the surface) are equal. In the mid and upper levels, if the temperature is within $5^{\circ}C$ of the dewpoint, that is a good indication that clouds exist at that level. Typically, clouds that form in the upper layer result from convection from the lower layer, where the sun warms the airmass and particles within the cloud are again released in the atmosphere and condensate in the upper layer.

5.3.4 Parallel Implementation

Our simplified simulation method is entirely implemented on the CPU. Results show that we achieve real-time simulation for convective clouds of about 250,000 particles maximum; the same applies to orographic clouds, though we have also to consider the further calculations resulting from the terrain influence in cloud formation. It is clear that parallel computing (e.g., using CUDA architecture from Nvidia) would greatly speed up the computations, so that we could simulate the behaviour of more complex cloud systems with millions of particles.

5.3.5 Other Natural Phenomena

Clouds are the most important phenomenon that occurs in the atmosphere. Rain and snow are other two atmospheric phenomena that can be predicted by thermodynamic diagrams. Interestingly, rain and snow are tied to clouds, but these phenomena are usually approached separately in computer graphics literature.

In the atmosphere, there are three cases when it comes the time to determine the type of precipitation:

- temperature less than 0 °C;
- only one freezing level;
- at least two freezing levels.

In the first case, two scenarios may occur: (i) the ambient temperature curve never goes over 0 °C isotherm; (ii) the cloud is never colder than $-8^{\circ}C$. In the first scenario, snow falls, while in the second scenario it is likely to have rain fall.

In the second case, the existence of a single freezing level means that the ambient temperature curve crosses the 0 °C isotherm only once. In this case, the type of precipitation is determined by calculating the freezing point and the point P_{-200} in the same ambient temperature curve located 200 mb below the freezing point. If the temperature of P_{-200} is greater than 0 °C, then precipitation is in the form of rain. Otherwise, if the freezing level is below the 1200 m, precipitation is in the form of snow.

In case of having at least two freezing levels, the ambient temperature curve crosses the 0 °C isotherm at least twice. This means that the atmosphere will be stratified into

cold and warm layers. The type of precipitation is determined by the strength of the warm layer. If the temperature in this layer is greater than 3 °C, then the precipitation is in the form of rain. If the temperature in this layer is lower than 1 °C, it is likely to have snow. Finally, if the temperature in the warm layer is greater than 1 °C and less than 3 °C both situations may occur, that is, rain mixed with snow.

Given these theoretical assumptions, one can use SkewT/LogP diagrams to predict the likelihood of rain and snow occurrence, as well as to develop new methods to couple these two natural phenomena to cloud formation. Moreover, these diagrams can also be applied to the rendering of melting snow over surfaces, due to the temperature profiles available in SkewT/LogP diagrams. Recently, [MGG+10], used ambient temperature curves and dewpoint temperature to characterise the evolution of snow accumulation on the ground; more specifically, these curves are used to determine the likelihood of falling snow during the day.

Bibliography

[AE96] Oleg A. Alduchov and Robert E. Eskridge. Improved magnus form approximation of saturation vapor pressure. *Journal of Applied Meteorology*, 35(4):601–609, 1996. 61

[AH04] N. Adabala and C. E. Hughes. A parametric model for real-time flickering fire. In *Proceedings of Computer Animation and Social Agents* (CASA'04), Geneva, Switzerland, July 7-9. IEEE Press, 2004. 132

[Ale05] John Alex. *Hybrid Sketching: A New Middle Ground Between 2-D and 3-D*. PhD thesis, Massachusetts Institute of Technology, 2005. 37

[Ale15] Institute Alexandra. Elementacular plugin. `http://elementacular.alexandra.dk`, September 2015. 7, 9, 11, 72, 98

[APKG07] Bart Adams, Mark Pauly, Richard Keiser, and Leonidas J. Guibas. Adaptively sampled particle fluids. *ACM Transactions on Graphics*, 26(3):48, 2007. 24

[AWSH61] Scott Airforce Base Air Weather Service HQ. *The use of the Skew-T/LogP Diagram in Analysis and Forecasting*, 1961. 49, 59

[Bli82] James F. Blinn. A generalization of algebraic surface drawing. *ACM Transactions on Graphics*, 1(3):235–256, 1982. 33

[BN04] Antoine Bouthors and Fabrice Neyret. Modeling clouds shape. In *Proceedings of Eurographics Short Papers* (EG'04), Genoble, France, August 30 - September 3. Eurographics Association, 2004. 18, 28, 29, 31, 44, 52

[BNL06] Antoine Bouthors, Fabrice Neyret, and Sylvain Lefebvre. Real-time realistic illumination and shading of stratiform clouds. In *Proceedings of the 2nd Eurographics Workshop on Natural Phenomena*, (NPH'06), Vienna, Austria, September 5, pages 41–50. Eurographics Association, 2006. 34, 36, 44, 85

[BNM+08] Antoine Bouthors, Fabrice Neyret, Nelson Max, Eric Bruneton, and Cyril Crassin. Interactive multiple anisotropic scattering in clouds. In *Proceedings of the 2008 Symposium on Interactive 3D Graphics and Games* (I3D'08), Redwood City, California, USA, February 15-17, pages 173–182. ACM Press, 2008. 18, 34, 36, 43, 44, 137

[BWDY15] Ferreira Barbosa, Charles Welton, Yoshinori Dobashi, and Tsuyoshi Yamamoto. Adaptive cloud simulation using position based fluids. *Computer Animation and Virtual Worlds*, 26(3-4):367–375, 2015. 18, 22, 23, 24, 42, 44

[BWK03] David Baraff, Andrew Witkin, and Michael Kass. Untangling cloth. *ACM Transactions on Graphics*, 22(3):862–870, 2003. 85

[CBK11] Yunus A Cengel, Michael A Boles, and Mehmet Kanoğlu. *Thermodynamics: an Engineering Approach*, volume 5. McGraw-Hill New York, 2011. 54, 133, 134

[CG12] Cyril Crassin and Simon Green. Octree-based sparse voxelization using the GNU hardware rasterizer. In *OpenGL Insights*, chapter 22, pages 303–318. CRC Press, 2012. 72

[CNLE09] Cyril Crassin, Fabrice Neyret, Sylvain Lefebvre, and Elmar Eisemann. Gigavoxels: Ray-guided streaming for efficient and detailed voxel rendering. In *Proceedings of the Symposium on Interactive 3D Graphics and Games* (I3D'09), Boston, Massachusetts, USA, February 27-March 1, pages 15–22. ACM Press, 2009. 35

[Cro00] Benjamin Crowell. *Newtonian Physics*, volume 1. Light and Matter, 2000. 54, 134

[CSI09] Deukhyun Cha, Sungjin Son, and Insung Ihm. GPU-assisted high quality particle rendering. *Computer Graphics Forum*, 28(4):1247–1255, 2009. 18, 28, 29, 30, 31, 44, 45

[CSYB05] Nicu D. Cornea, Deborah Silver, Xiaosong Yuan, and Raman Balasubramanian. Computing hierarchical curve-skeletons of 3D objects. *The Visual Computer*, 21(11):945–955, 2005. 32

[CTM08] Sean Curtis, Rasmus Tamstorf, and Dinesh Manocha. Fast collision detection for deformable models using representative-triangles. In *Proceedings of the 2008 Symposium on Interactive 3D Graphics and Games* (I3D'08), Redwood City, California, USA, February 15-17, pages 61–69. ACM Press, 2008. 85

[DBLR12] Joo-Young Do, Nakhoon Baek, Chang-Woo Lee, and Kwan-Woo Ryu. A semi-automatic generation method for visually-plausible virtual clouds. *International Journal of Multimedia and Ubiquitous Engineering*, 7(2):135–146, 2012. 29, 30, 31, 44

[DBS+07] Finale Doshi, Emma Brunskill, Alexander Shkolnik, Thomas Kollar, Khashayar Rohanimanesh, Russ Tedrake, and Nicholas Roy. Collision detection in legged locomotion using supervised learning. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems* (IROS'07), San Diego, California, United States, October 29-November 2, pages 317–322. IEEE Computer Society, 2007. 85

[DCGG11] Emmanuelle Darles, Benoît Crespin, Djamchid Ghazanfarpour, and Jean-Christophe Gonzato. A survey of ocean simulation and rendering techniques in computer graphics. *Computer Graphics Forum*, 30(1):43–60, 2011. 18, 19

[DFS05] Neil Dodgson, Michael S. Floater, and Malcolm Sabin. *Advances in Multiresolution for Geometric Modelling*. Springer Science & Business Media, 2005. 46

[DIO+12] Yoshinori Dobashi, Wataru Iwasaki, Ayumi Ono, Tsuyoshi Yamamoto, Yonghao Yue, and Tomoyuki Nishita. An inverse problem approach for automatically adjusting the parameters for rendering clouds using photographs. *ACM Transactions on Graphics*, 31(6):145, 2012. 39, 40, 44

[DKNY08] Yoshinori Dobashi, Katsutoshi Kusumoto, Tomoyuki Nishita, and Tsuyoshi Yamamoto. Feedback control of cumuliform cloud formation based on computational fluid dynamics. *ACM Transactions on Graphics*, 27(3):94, 2008. 18, 22, 23, 24, 42, 44, 81, 109

[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), New Orleans, Louisiana, USA, July 23-28, pages 19–28. ACM Press/Addison-Wesley Publishing Co., 2000. 18, 25, 26, 44, 52, 137

[DNO98] Yoshinori Dobashi, Tomoyuki Nishita, and Tsuyoshi Okita. Animation of clouds using cellular automaton. In *Proceedings of Computer Graphics and Imaging* (CGI'98), Halifax, Canada, June 1-4, pages 251–256. IEEE Computer Society, 1998. 18, 25, 26, 44

[DNYO98] Yoshinori Dobashi, Tomoyuki Nishita, Hideo Yamashita, and Tsuyoshi Okita. Modeling of clouds from satellite images using metaballs. In *Proceedings of the 6th Pacific Conference on Computer Graphics and Applications* (PG'98), Singapure, October 26-28, pages 53–60. IEEE Computer Society, 1998. 18, 34, 38, 39, 44, 45

[DSY10] Yoshinori Dobashi, Yusuke Shinzo, and Tsuyoshi Yamamoto. Modeling of clouds from a single photograph. *Computer Graphics Forum*, 29(7):2083–2090, 2010. 18, 39, 40, 44

[DTM96] Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques* (SIGGRAPH '96), New Orleans, LA, USA, August 04-09, pages 11–20. ACM Press, 1996. 38

[DYN06] Yoshinori Dobashi, Tsuyoshi Yamamoto, and Tomoyuki Nishita. A controllable method for animation of Earth-scale clouds. In *Proceedings of Computer Animation and Social Agents* (CASA'06), Geneva, Switzerland, July 5-7, pages 43–52. John Wiley and Sons Ltd, 2006. 22, 23, 44

[Ebe04] David S. Ebert. Interactive cloud modeling and photorealistic atmospheric rendering. In *ACM SIGGRAPH 2004 Course Notes: The elements of Nature Interactive and Realistic Techniques, Section 6* (SIGGRAPH'04), Los Angeles, California, USA, August 8-12. ACM Press, 2004. 33

[Ema94] Kerry A. Emanuel. *Atmospheric Convection*. Oxford University Press, 1994. 76

[EMF02] Douglas Enright, Stephen Marschner, and Ronald Fedkiw. Animation and rendering of complex water surfaces. *ACM Transactions on Graphics*, 21(3):736–744, 2002. 16

[EMP+03] David S Ebert, F Kenton Musgrave, Darwyn Peachey, Ken Perlin, and Steve Worley. *Texturing and Modeling: A Procedural Approach*. Morgan Kaufmann, 3rd edition, 2003. 18, 33, 35, 43, 44, 52

[Eri04] Christer Ericson. *Real-time Collision Detection*. CRC Press, 2004. 85

[ES00] Pantelis Elinas and Wolfgang Stürzlinger. Real-time rendering of 3D clouds. *Journal of Graphic Tools*, 5:33–45, 2000. 33, 44, 52

[FOK05] Bryan E. Feldman, James F. O'Brien, and Bryan M. Klingner. Animating gases with hybrid meshes. *ACM Transactions on Graphics*, 24(3):904–909, 2005. 133

[FSJ01] Ronald Fedkiw, Jos Stam, and Henrik Wann Jensen. Visual simulation of smoke. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques* (SIGGRAPH'01), Los Angeles, California, USA, August 12-17, pages 15–22. ACM Press, 2001. 20, 21

[Gar85] Geoffrey Y. Gardner. Visual simulation of clouds. *ACM SIGGRAPH Computer Graphics*, 19:297–304, 1985. 3, 33, 34, 35, 44, 52

[GD04] Stéphane Guy and Gilles Debunne. Monte-Carlo Collision Detection. Technical Report RR-5136, INRIA, March 2004. Available from: https://hal.inria.fr/inria-00071447. 85

[GD07] Jakub Grudzinski and Adrian Debowski. Clouds and atmospheric phenomena simulation in real-time 3D graphics. In *Proceedings of the 3rd International Conference on Computer Vision/Computer Graphics Collaboration Techniques* (MIRAGE'07), Rocquencourt, France, March 28-30, volume 4418 of *Lecture Notes in Computer Science*, pages 117–127. Springer Verlag, 2007. 28, 44, 51, 52

[Get99] A.V. Getling. *Rayleigh-Bénard convection: Structures and Dynamics*, volume 11 of *Advanced Series in Non-Linear Dynamics*. World Scientific, 1999. 108

[Gre05] S. Green. Implementing Improved Perlin Noise. In *GPU Gems 2*, chapter 26, pages 409–416. Addison-Wesley Professional, 2005. 33, 52

[GRWS04] Robert Geist, Karl Rasche, James Westall, and Robert Schalkoff. Lattice-Boltzmann Lighting. In *Proceedings of the 15th Eurographics Conference on Rendering Techniques* (EGSR'04), Norkooping, Sweden, June 21-23, pages 355–362. ACM Press, 2004. 21, 23, 44

[GSW07] Robert Geist, Jay Steele, and James Westall. Convective clouds. In *Eurographics Workshop on Natural Phenomena* (NPH'07), Prague, Czech Republic, September 4, pages 23–30. The Eurographics Association, 2007. 18, 21, 23, 52, 137

[GVJ+09] Abel Gomes, Irina Voiculescu, Joaquim Jorge, Brian Wyvill, and Callum Galbraith. *Implicit Curves and Surfaces: Mathematics, Data Structures and Algorithms*. Springer Verlag, 2009. 32

[Ham47] William Rowan Hamilton. The hodograph, or a new method of expressing in symbolical language the newtonian law of attraction. In *Proceedings of the Royal Irish Academy*, volume 3, pages 344–353, 1847. 55

[HBSL03] Mark J. Harris, William V. Baxter, Thorsten Scheuermann, and Anselmo Lastra. Simulation of cloud dynamics on graphics hardware. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware* (HWWS'03), San Diego, California, July 26-27, pages 92–101. Eurographics Association, 2003. 18, 20, 23, 44, 52

[HGH15] Zhanpeng Huang, Guanghong Gong, and Liang Han. Physically-based smoke simulation for computer graphics: a survey. *Multimedia Tools and Applications*, 74(18):7569–7594, 2015. 18, 19

[HH12] Roland Hufnagel and Martin Held. A survey of cloud lighting and rendering techniques. *Journal of WSCG*, 20:205–215, 2012. 18, 19, 43

[HHS07] Roland Hufnagel, Martin Held, and Florian Schröder. Large-scale, realistic cloud visualization based on weather forecast data. In *Proceedings of the 9th IASTED International Conference on Computer Graphics and Imaging* (CGIM'07), Innsbruck, Austria, February 13-15, pages 54–59. ACTA Press, 2007. 34, 44

[HKK07] Takahiro Harada, Seiichi Koshizuka, and Yoichiro Kawaguchi. Smoothed particle hydrodynamics in complex shapes. In *Proceedings of the 23rd Spring Conference on Computer Graphics* (SCCG'07), Budmerice, Slovakia, April 26-28, pages 191–197. ACM Press, 2007. 86

[HL01] Mark J. Harris and Anselmo Lastra. Real-time cloud rendering. *Computer Graphics Forum*, 20(3):76–85, 2001. 29, 30, 41, 44, 72, 137

[HL02] Mark J. Harris and Anselmo Lastra. Real-time cloud rendering for games. In *Proceedings of Game Developers Conference* (GDC'02), San Francisco, California, March 5—7, pages 21–29, 2002. 29, 72

[JTT01] P. Jiménez, F. Thomas, and C. Torras. 3D collision detection: a survey. *Computers & Graphics*, 25(2):269–285, 2001. 85

[KAD+06] Richard Keiser, Bart Adams, Philip Dutré, Leonidas Guibas, and Mark Pauly. Multiresolution particle-based fluids. Technical Report 520, Department of Computer Science, ETH Zurich, 2006. 46

[KPH+03] Joe Kniss, Simon Premoze, Charles Hansen, Peter Shirley, and Allen McPherson. A model for volume lighting and modeling. *IEEE Transactions on Visualization and Computer Graphics*, 9(2):150–162, 2003. 33, 35, 43, 44, 85

[KPS+14] David Koerner, Jamie Portsmouth, Filip Sadlo, Thomas Ertl, and Bernd Eberhardt. Flux-limited diffusion for multiple scattering in participating media. *Computer Graphics Forum*, 33(6):178–189, 2014. 72

[KPTB99] Evaggelia-Aggeliki Karabassi, Georgios Papaioannou, Theoharis Theoharis, and Alexander Boehm. Intersection test for collision detection in particle systems. *Journal of Graphics Tools*, 4(1):25–37, 1999. 86

[KVH84] James T Kajiya and Brian P. Von Herzen. Ray tracing volume densities. In *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques* (SIGGRAPH'84), Minneapolis, USA, July 23-27, volume 18, pages 165–174. ACM Press, 1984. 20, 21, 23, 44

[Law05] Mark G. Lawrence. The relationship between relative humidity and the dewpoint temperature in moist air: A simple conversion and applications. *Bulletin of the American Meteorological Society*, 86(2):225–233, 2005. 61

[LCN99] Jean-Christophe Lombardo, Marie-Paule Cani, and Fabrice Neyret. Real-time collision detection for virtual surgery. In *Proceedings of the Computer Animation* (CA'99), Geneve, Switzerland, May 26-29, pages 82–90. IEEE Computer Society, 1999. 85

[LGS06] Florian Levet, Xavier Granier, and Christophe Schlick. Fast sampling of implicit surfaces by particle systems. In *Proceedings of the IEEE International Conference on Shape Modeling and Applications* (SMI'06), Matsushima, Japan, June 14-16, pages 39–39. IEEE Computer Society, 2006. 30

[LJW06] Shengjun Liu, Xiaogang Jin, and Charlie CL Wang. Target shape controlled cloud animation. In *Advances in Computer Graphics*, pages 578–585. Springer, 2006. 36, 37, 44

[LKS+96] Craig Lee, Carl Kesselman, Stephen Schwab, et al. Near-real-time satellite image processing: Metacomputing in cc++. *IEEE Computer Graphics and Applications*, 16(4):79–84, 1996. 38, 44

[LWGP07] Shiguang Liu, Zhangye Wang, Zheng Gong, and Qunsheng Peng. Real time simulation of a tornado. *The Visual Computer*, 23(8):559–567, 2007. 22

[LWGP08] Shiguang Liu, Zhangye Wang, Zheng Gong, and Qunsheng Peng. Simulation of atmospheric binary mixtures based on two-fluid model. *Graphical Models*, 70(6):117–124, 2008. 22

[LY15] Xiaohui Liang and Chunqiang Yuan. Derivation of 3D cloud animation from geostationary satellite images. *Multimedia Tools and Applications*, pages 1–21, 2015. 38, 39, 44

[Man06] P. Man. Generating and real-time rendering of clouds. In *Proceedings of the 10th Central European Seminar on Computer Graphics* (CESCG'06), in conjunction with Spring Conference on Computer Graphics (SCCG'06), Vienna, Austria, April 24, 2006. 52

[Mas57] Basil John Mason. *The Physics of Clouds*. Oxford University Press, 1957. 13, 52, 86

[MDCN03] Ryoichi Mizuno, Yoshinori Dobashi, Bing-Yu Chen, and Tomoyuki Nishita. Physics motivated modeling of volcanic clouds as a two fluids model. In *Proceedings of the 11th Pacific Conference on Computer Graphics and Applications* (PG'03), Canmore, Alberta, Canada, October 8-10, pages 440–444. IEEE Computer Society, 2003. 21

[MDN02] Ryo Miyazaki, Yoshinori Dobashi, and Tomoyuki Nishita. Simulation of cumuliform clouds based on computational fluid dynamics. In *Proceedings of Eurographics 2002 Short Papers* (EG'02), Saarbrucken, Germany, September 2-6, pages 405–410. Eurographics Association, 2002. 20, 21, 22, 23, 24, 42, 44

[MDN04] Ryoichi Mizuno, Yoshinori Dobashi, and Tomoyuki Nishita. Modeling of volcanic clouds using CML. *Journal of Information Science and Engineering*, 20(2):219–232, 2004. 21, 44

[MGG+10] N. Maréchal, E. Guérin, E. Galin, S. Mérillou, and N. Mérillou. Heat transfer simulation for modeling realistic winter sceneries. *Computer Graphics Forum*, 29(2):449–458, 2010. 45, 64, 111

[MHHR07] Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. Position based dynamics. *Journal of Visual Communication and Image Representation*, 18(2):109–118, 2007. 23

[MKE03] Johannes Mezger, Stefan Kimmerle, and Olaf Etzmu. Hierarchical techniques in collision detection for cloth animation. *Journal of WSCG*, 11:1–3, 2003. 85

[MLH+15] James Murphy, Colin Levy, Aaron Hartline, Austin Lee, Dirk Van Gelder, Byron Bashforth, and Farhez Rayani. Disney Pixar's "LAVA": Moving Mountains. In *ACM SIGGRAPH 2015 Computer Animation Festival* (SIGGRAPH'15), Los Angeles, California, USA, August 9-13, pages 170–170. ACM Press, 2015. 85

[MM06] Cedric S. Ma and Robert H. Miller. MILP optimal path planning for real-time applications. In *Proceedings of the American Control Conference* (ACC'06), Minneapolis, Minesota, USA, June 14-16, pages 4945–4950. IEEE Computer Society, 2006. 85

[MM13] Miles Macklin and Matthias Müller. Position based fluids. *ACM Transactions on Graphics*, 32(4):104, 2013. 22

[MMPZ12] B. Miller, Ken Museth, D. Penney, and N.F. Zafar. Cloud modeling and rendering for "puss in boots". *ACM SIGGRAPH Talk*, 2012. 18, 35, 36, 44, 52

[MMS04] Viorel Mihalef, Dimitris Metaxas, and Mark Sussman. Animation and control of breaking waves. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (SCA'04), Grenoble, France, August 27-29, pages 315–324. Eurographics Association, 2004. 132

[MST+04] Matthias Müller, Simon Schirm, Matthias Teschner, Bruno Heidelberger, and Markus Gross. Interaction of fluids with deformable solids. *Computer Animation and Virtual Worlds*, 15(3-4):159–171, 2004. 85, 86

[Mus13] Ken Museth. VDB: High-resolution sparse volumes with dynamic topology. *ACM Transactions on Graphics*, 32(3):27:1–27:22, 2013. 35, 52

[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), Tokyo, Japan, October 16-18, pages 363–372. IEEE Computer Society, 2001. 18, 20, 21, 22, 23, 44, 52

[NDN96] Tomoyuki Nishita, Yoshinori Dobashi, and Eihachiro Nakamae. Display of clouds taking into account multiple anisotropic scattering and sky light. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques* (SIGGRAPH '96), New Orleans, Louisiana, USA, August 4-9, pages 379–386. ACM Press, 1996. 33, 44

[Ney97] Fabrice Neyret. Qualitative simulation of convective cloud formation and evolution. In *Proceedings of the Eurographics Workshop on Computer Animation and Simulation* (EWAS'97) Budapest, Hungary, September 2-3, pages 113–124. Springer, 1997. 18, 28, 34, 51, 52

[Nor38] Normand. On instability from water vapour. *Quarterly Journal of the Royal Meteorological Society*, 64(273):47–70, 1938. 14, 87

[NR92] Kai Nagel and Ehrhard Raschke. Self-organizing criticality in cloud formation? *Physica A: Statistical Mechanics and its Applications*, 182(4):519–531, 1992. 21, 25

[oCoDoT72] United States. Dept. of Commerce, United States. Dept. of Defense, and United States. Dept. of Transportation. *Radiosonde code: Standards and Procedures for the Coding of Radiosonde Reports, effective January 1, 1972*. Federal Meteorological Handbook. U.S. Dept. of Commerce, U.S. Dept. of Defense, and U.S. Dept. of Transportation; for sale by Supt. Docs., U.S. Govt. Print. Office, 1972. 61

[OMK02] Derek Overby, Zeki Melek, and John Keyser. Interactive physically-based cloud simulation. In *Proceedings of the 10th Pacific Conference on Computer Graphics and Applications* (PG'02), Tsingua, Beijing, October 9-11, page 469–470. IEEE Computer Society, 2002. 20, 21, 23, 44

[Pea02] Robert Penrose Pearce. *Meteorology at the Millennium*, volume 83. Academic Press, 2002. 93

[Per85] Ken Perlin. An image synthesizer. *ACM SIGGRAPH Computer Graphics*, 19(3):287–296, 1985. 33, 52

[PG07] Renato Pajarola and Enrico Gobbetti. Survey of semi-regular multiresolution models for interactive terrain rendering. *The Visual Computer*, 23(8):583-605, 2007. 46

[PPLT09] Afonso Paiva, Fabiano Petronetto, Thomas Lewiner, and Geovan Tavares. Particle-based viscoplastic fluid/solid simulation. *Computer-Aided Design*, 41(4):306–314, 2009. 85, 86

[PTB+03] Simon Premoze, Tolga Tasdizen, James Bigler, Aaron Lefohn, and Ross T. Whitaker. Particle-based simulation of fluids. *Computer Graphics Forum*, 22(3):401–410, 2003. 132

[PVGV+04] Marc Pollefeys, Luc Van Gool, Maarten Vergauwen, Frank Verbiest, Kurt Cornelis, Jan Tops, and Reinhard Koch. Visual modeling with a hand-held camera. *International Journal of Computer Vision*, 59(3):207–232, 2004. 38

[Qua10] Long Quan. *Image-Based Modeling*. Springer Publishing Company, Incorporated, 1st edition, 2010. 38

[Ree83] W. T. Reeves. Particle systems: a technique for modeling a class of fuzzy objects. *ACM Transactions on Graphics*, 2(2):91–108, 1983. xv, 27

[REHL03] Kirk Riley, David Ebert, Charles Hansen, and Jason Levit. Visually accurate multi-field weather visualization. In *Proceedings of the 14th IEEE Visualization* (VIS'03), Seattle, Washington, October 19-24, pages 37–44. IEEE Computer Society, 2003. 18, 31, 32, 44, 45

[RGF+04] Laks Raghupathi, Laurent Grisoni, François Faure, Damien Marchal, Marie-Paule Cani, and Christophe Chaillou. An intestinal surgery simulator: Real-time collision processing and visualization. *IEEE Transactions on Visualization and Computer Graphics*, 10(6):708–718, 2004. 85

[RH02] Arthur Richards and Jonathan P. How. Aircraft trajectory planning with collision avoidance using mixed integer linear programming. In *Proceedings of the American Control Conference* (ACC'02), Anchorage, Alaska, USA, May 8-10, volume 3, pages 1936–1941. IEEE Computer Society, 2002. 85

[RNGF03] Nick Rasmussen, Duc Quang Nguyen, Willi Geiger, and Ronald Fedkiw. Smoke simulation for large scale phenomena. In *Proceedings of the ACM SIGGRAPH Conference* (SIGGRAPH'03), San Diego, California, USA, July 27-31, pages 703–707. ACM Press, 2003. 133

[RSK+06] Kirk Riley, Yuyan Song, Martin Kraus, David S. Ebert, and Jason J. Levit. Visualization of structured nonuniform grids. *IEEE Computer Graphics and Applications*, 26(1):46–55, 2006. 34, 44, 45

[RY89] R.R. Rogers and M.K. Yau. *A Short Course in Cloud Physics*. International Series in Natural Philosophy. Butterworth Heinemann, Burlington, MA, 3 edition, 1989. 54, 63, 134

[SAC+99] Dan Stora, Pierre-Olivier Agliati, Marie-Paule Cani, Fabrice Neyret, and Jean-Dominique Gascuel. Animating lava flows. In *Proceedings of the Conference on Graphics Interface* (GI'99), Kingston, Ontario, Canada, June 2-4, pages 203–210. Morgan Kaufmann Publishers Inc., 1999. 85, 86

[SG04] Frutuoso G.M. Silva and Abel J.P. Gomes. Normal-based simplification algorithm for meshes. In *Proceedings of Theory and Practice of Computer Graphics* (TPCG'04), Bournemouth, England, June 10-10, pages 211–218. IEEE Computer Society, 2004. 95

[SGPJ12] Jerôme Schalkwijk, Eric J. Griffith, Frits H. Post, and Harm J.J. Jonker. High-performance simulations of turbulent clouds on a desktop PC. *Bulletin of the American Meteorological Society*, 93(3):307–314, 2012. 6

[SH08] Zhang Shengxiang and Pei Hailong. Real-time optimal trajectory planning with terrain avoidance using milp. In *Proceedings of the 2nd International*

Symposium on Systems and Control in Aerospace and Astronautics (ISSCAA'2008), Shenzhen, China, December 10-12, pages 1–5. IEEE Press, 2008. 85

[SK99] Milos Sramek and Arie E. Kaufman. Alias-free voxelization of geometric objects. *IEEE Transactions on Visualization and Computer Graphics*, 5(3):251–267, 1999. 32

[Sof15] Sundog Software. Silverlining. http://www.sundog-sof.com, December 2015. 42

[SSEH03] Joshua Schpok, Joseph Simons, David S. Ebert, and Charles Hansen. A real-time cloud modeling, rendering, and animation system. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (SCA'03), San Diego, California, USA, July 26-27, pages 160–166. Eurographics Association, 2003. 18, 34, 35, 43, 44, 52, 137

[Sta99] Jos Stam. Stable fluids. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques* (SIGGRAPH'99), Los Angeles, California, USA, August 8-13, pages 121–128. ACM Press/Addison-Wesley Publishing Co., 1999. 20, 52, 132

[SU94] John Steinhoff and David Underhill. Modification of the euler equations for "vorticity confinement": Application to the computation of interacting vortex rings. *Physics of Fluids*, 6(8):2738–2744, 1994. 133

[SY05] Lin Shi and Yizhou Yu. Controllable smoke animation with guiding objects. *ACM Transactions on Graphics*, 24(1):140–164, 2005. 37

[SYL08] Helong Shen, Yong Yin, and Yongjin Li. Real-time dynamic simulation of 3D cloud for marine search and rescue simulator. In *Proceedings of the 7th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry* (VRCAI'08), Singapore, December 8-9, pages 14:1–14:5. ACM Press, 2008. 26, 44

[SYS+06] Yuyan Song, Jing Ye, Nikolai Svakhine, Sonia Lasher-Trapp, Mike Baldwin, and David Ebert. An atmospheric visual analysis and exploration system. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1157–1164, 2006. 34, 36, 44, 45

[TBV02] Andrzej Trembilski, Andreas Broßler, and Abt Visualisierung Und Virtuelle. Surface-based efficient cloud visualisation for animation applications. In *Proceedings of the 10th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision* (WSCG'02), Plzen, Czech Republic, February 4-8, page 453—461, 2002. 3, 34, 36, 44, 45, 52, 137

[THS+15] Natalya Tatarchuk, Sébastien Hillaire, Tomasz Stachowiak, Huw Bowles, Beibei Wang, Ari Silvennoinen, and Ville Timonen. Advances in Real Time Rendering, Part I. In *ACM SIGGRAPH 2015 Courses* (SIGGRAPH'15), Los Angeles, California, USA, August 9-13. ACM Press, 2015. 17

[TKH+05] M. Teschner, S. Kimmerle, B. Heidelberger, G. Zachmann, L. Raghupathi, A. Fuhrmann, M.-P. Cani, F. Faure, N. Magnenat-Thalmann, W. Strasser, and P. Volino. Collision detection for deformable objects. *Computer Graphics Forum*, 24(1):61–81, 2005. 85

[TY09] Jie Tan and XuBo Yang. Physically-based fluid animation: A survey. *Science in China Series F: Information Sciences*, 52(5):723–740, 2009. 18, 19

[US10] Eric M. Upchurch and Sudhanshu K. Semwal. Dynamic cloud simulation using cellular automata and texture splatting. In *Proceedings of the 2010 Summer Computer Simulation Conference* (SCSC'10), Otawa, Ontario, Canada, July 11-15, pages 270–277. Society for Computer Simulation International, 2010. 26, 44, 85

[USK05] Tamás Umenhoffer and László Szirmay-Kalos. Real-time rendering of cloudy natural phenomena with hierarchical depth impostors. In *Proceedings of Eurographics Short Papers* (EG'05), Dublin, Ireland, August 29- September 02. Eurographics Association, 2005. 30, 31, 44

[Vos85] R.F. Voss. Random fractal forgeries. In *Fundamental Algorithms for Computer Graphics*, volume 17 of *NATO Advanced Study Institute*, pages 805–835. Springer-Verlag, 1985. 33, 52

[VT00] Pascal Volino and Nadia Magnenat Thalmann. Implementing fast cloth simulation with collision response. In *Proceedings of the Computer Graphics International Conference* (CGI'00), Geneva, Switzerland, June 19-24, page 257. IEEE Computer Society, 2000. 85

[Wag87] David H. Wagner. Equivalence of the Euler and Lagrangian equations of gas dynamics for weak solutions. *Journal of Differential Equations*, 68(1):118–136, 1987. 132

[Wan03] Niniane Wang. Realistic and fast cloud rendering in computer games. In *ACM SIGGRAPH 2003 Sketches & Applications* (SIGGRAPH'03), San Diego, California, USA, July 27-31, pages 1–1. ACM Press, 2003. 29, 31, 44

[Wan04] Niniane Wang. Realistic and fast cloud rendering. *Journal of Graphics Tools*, 9(3):21–40, 2004. 29

[WBC08] Jamie Wither, Antoine Bouthors, and Marie-Paule Cani. Rapid sketch modeling of clouds. In *Proceedings of the 5th Eurographics Conference on Sketch-Based Interfaces and Modeling* (SBIM'08), Annecy, France, June 11-13, pages 113–118. Eurographics Association, 2008. 36, 37, 44, 52

[WPKK07] Be Wang, Jingliang Peng, Youngmin Kwak, and C-C Jay Kuo. Efficient and realistic cumulus cloud simulation based on similarity approach. In *Advances in Visual Computing*, volume 4841 of *Lecture Notes in Computer Science*, pages 781–791. Springer, 2007. 18, 22, 23, 44

[Wre12] Magnus Wrenninge. *Production Volume Rendering: Design and Implementation*. CRC Press, 2012. 72

[WSYM12] Wang Wenke, Li Sikun, Guo Yumeng, and Xiong Min. Automatic generation of large scale 3D clouds based on weather forecast data. In *Proceedings of the 2012 International Conference on Virtual Reality and Visualization* (ICVRV'12), Qinhuangdao, China, September 14-15, pages 69–73. IEEE Computer Society, 2012. 31, 32, 44, 45, 52

[WT90] G. Wyvill and A. Trotman. Ray-tracing soft objects. In *Proceedings of the 8th Computer Graphics International* (CGI'90), Singapore, June 25-29, pages 469–476. Springer-Verlag New York, Inc., 1990. 39

[YK97] Tatsuo Yanagita and Kunihiko Kaneko. Modeling and characterization of cloud dynamics. *Physical Review Letters*, 78(22):4297, 1997. 21

[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. 18, 40, 44

[YLHY13] Chunqiang Yuan, Xiaohui Liang, Shiyu Hao, and Guang Yang. Modeling large scale clouds from satellite images. In *Proceedings of the Pacific Graphics Short Papers presentation* (PG'13), Singapore, Tailand, October 7-9. The Eurographics Association, 2013. 18, 38, 39, 44

[Yus14] Egor Yusov. High-performance rendering of realistic cumulus clouds using pre-computed lighting. In *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on High Performance Graphics* (HPG'14), Lyon, France, June 23-25, pages 127–136. ACM Press, 2014. 28, 31, 32, 44, 45

[YW+11] Chung-Min Yu, Chung-Ming Wang, et al. An effective framework for cloud modeling, rendering, and morphing. *Journal of Information Science and Engineering*, 27(3):891–913, 2011. 30, 31, 44

[ZZW11] Wu ZhaoHui, Zhou Zhong, and Wu Wei. Realistic fire simulation: a survey. In *Proceedings of the 12th International Conference on Computer-Aided Design and Computer Graphics* (CAD/Graphics'11), Jinan, China, September 15-17, pages 333–340. IEEE Computer Society, 2011. 18, 19

Appendix A

Types of Clouds

This appendix contains images that illustrate the ten main sorts of clouds that form in the atmosphere, as illustrated in Figures A.1-A.2. In this thesis, we focus on cumulus clouds, which are the most common clouds in the atmosphere.

(a) Cumulus. FischX / Wiki Commons.

(b) Stratus. Piccolo Namek / Wiki Commons.

(c) Stratocumulus. LivingShadow / Wiki Commons.

(d) Cumulonimbus. Fir0002 / Wiki Commons.

Figure A.1: The first four out of ten main cloud types that form in the atmosphere.

(a) Cirrus. ⓒⓘⓞPrzemyslaw Idzkiewicz / Wiki Commons.

(b) Cirrocumulus. ⓒⓘⓞSimon Eugster / Wiki Commons.

(c) Cirrostratus. ⓒⓘⓞSimon Eugster / Wiki Commons.

(d) Altocumulus. ⓒⓘⓞFir0002 /Wiki Commons.

(e) Altostratus. ⓒⓘⓞThe Great Cloudwatcher / Wiki Commons.

(f) Nimbostratus. ⓒⓘⓞLiving Shadow / Wiki Commons.

Figure A.2: The last six out of ten main cloud types that form in the atmosphere.

Appendix B

Background on Physics

B.1 Equations of Fluids

Many natural phenomena are characterized as fluids, with their dynamics governed by the incompressible Navier-Stokes equations, which are usually written as:

$$\frac{\partial \vec{u}}{\partial t} + \underbrace{\vec{u} \cdot \nabla \vec{u}}_{\substack{\text{convective} \\ \text{term}}} + \underbrace{\frac{1}{\rho} \nabla p}_{\substack{\text{pressure} \\ \text{term}}} = \underbrace{\vec{g}}_{\substack{\text{external} \\ \text{acceleration}}} + \underbrace{\nu \nabla \cdot \nabla \vec{u}}_{\substack{\text{viscosity} \\ \text{term}}} \tag{B.1}$$

$$\nabla \cdot \vec{u} = 0 \tag{B.2}$$

where $\vec{u}$ stands for the velocity of the fluid, ρ the density of the fluid, p the pressure (i.e., the force per unit area that the fluid exerts on any contact body), $\vec{g}$ the acceleration due to gravity, which usually is (0,-9.81,0) m/s^2, and ν denotes the kinematic viscosity, which measures how viscous the fluid is. Fluids like molasses have high viscosity, and fluids like alcohol have low viscosity. In other words, viscosity measures how much the fluid resists deforming while it flows.

B.1.1 The Momentum Equation

Equation (B.1) is called the momentum equation. It is a vector equation that refers to the acceleration of the fluid due to the forces acting on it.

In order to understand this equation, lets consider a fluid simulation using a particle system. Each particle would have a mass m, a volume V and a velocity $\vec{u}$. To integrate the system forward in time, we use Newton's equation $\vec{F} = m\vec{a}$ that defines the forces acting on each particle, which make it to move. The acceleration of a given particle can be written as

$$\vec{a} = \frac{D\vec{u}}{Dt} \tag{B.3}$$

where D is called the material derivative[1]. Newton's Law is now:

$$m\frac{D\vec{u}}{Dt} = \vec{F} \tag{B.5}$$

Now, referring to $\vec{F}$, the forces acting on the particle. The first to consider is gravity: $m\vec{g}$. The rest of the fluid also exerts forces on the particle.

The first of the fluid forces is the pressure. Such a force results from the pressure imbalance around each particle. This results in a force that points away from high pressures towards low pressures, which is expressed as $-\nabla p$. By integrating this over the volume of the fluid, one obtains the pressure force. For example, if the pressure is equally distributed in every direction, the resulting net force is zero.

The other fluid force is due to viscosity. Typically, a viscous fluid tries to resist deforming. This type of force makes a particle move at the average velocity of the nearby particles, which is given by the Laplacian $\nabla\cdot\nabla$. Considering these forces, the movement of the particle is given by:

$$m\frac{D\vec{u}}{Dt} = m\vec{g} - V\nabla p + V\mu\nabla\cdot\nabla\vec{u} \tag{B.6}$$

where μ stands for the dynamic viscosity coefficient.

Dividing Equation (B.6) by the volume V, and considering that $m/V = \rho$, we obtain

$$\rho\frac{D\vec{u}}{Dt} = \rho\vec{g} - \nabla p + \mu\nabla\cdot\nabla\vec{u} \tag{B.7}$$

Dividing Equation (B.7) by the density and rearranging the terms,

$$\frac{D\vec{u}}{Dt} + \frac{1}{\rho}\nabla p = \vec{g} + \frac{\mu}{\rho}\nabla\cdot\nabla\vec{u} \tag{B.8}$$

Considering that kinematic viscosity is given by $\nu = \mu/V$, Equation (B.8) results in

$$\frac{D\vec{u}}{Dt} + \frac{1}{\rho}\nabla p = \vec{g} + \nu\nabla\cdot\nabla\vec{u} \tag{B.9}$$

which is the momentum equation using material derivative D/Dt. In order to under-

[1]The derivative of a field with respect to a fixed position in space is called the Eulerian derivative, while the derivative following a moving parcel is called the convective or material derivative. The material derivative is defined as the operator:

$$\frac{D}{Dt} = \frac{\partial}{\partial t} + \vec{u}\cdot\nabla \tag{B.4}$$

stand what the material derivative is and how it relates to Equation (B.1) and Equation (B.2), it is necessary to understand the difference between the Lagrangian and Eulerian approaches for fluids (cf. Section B.2).

B.1.1.1 Incompressibility Condition

A fluid characterised by a "compressible flow" admits changes in its volume. Computing volume changes of a fluid is time-consuming, and its effect is negligible when a fluid moves at a macroscopic level (water sloshing, smoke billowing, etc.). This means that we can treat fluids as "incompressible".

Let us then consider the volume V of a fluid, Ω, and its boundary surface, $\partial\Omega$. By integrating the normal component $\hat{n}$ of the volume velocity around the boundary of the fluid, we are able to measure how fast the volume is changing as follows:

$$\frac{d}{dt}V(\Omega) = \partial\Omega\vec{u}\cdot\hat{n} \tag{B.10}$$

For an incompressible fluid, the volume is constant (the rate of change is zero), so Equation (B.10) can be expressed as follows:

$$\partial\Omega\vec{u}\cdot\hat{n} = 0 \tag{B.11}$$

Using the divergence theorem, Equation (B.11) changes to a volume integral, and we obtain

$$\Omega\nabla\cdot\vec{u} = 0 \tag{B.12}$$

Taking into consideration that the only function that integrates to zero —independently of the volume of integration— is the null function, the integrand has to be zero everywhere. So, we have:

$$\nabla\cdot\vec{u} = 0 \tag{B.13}$$

which is the incompressibility condition, i.e., the second Navier-Stokes equation (see Equation B.2).

B.1.1.2 Euler Equations

In computer graphics, viscosity plays a minor role, and thus the viscous term is negligible, so the simpler the equations, the better. Typical situations where viscosity is extremely important are honey simulation, melting and water droplets. Most numerical methods for simulating fluids unavoidably introduce errors which can be physically reinterpreted as viscosity. Therefore, by dropping viscosity, methods still generate

something that looks realistic and, in fact, one of the challenges in computational fluid dynamics is avoiding this viscosity errors as much as possible.

The Navier-Stokes equations without viscosity are called “Euler equations”, and such an ideal fluid with no viscosity is called “inviscid”. Therefore, dropping the term ν in Equation (B.9), we obtain

$$\frac{D\vec{u}}{Dt} + \frac{1}{\rho}\nabla p = \vec{g} \tag{B.14}$$

$$\nabla \cdot \vec{u} = 0 \tag{B.15}$$

B.2 Computational Physics

Physically-based fluid simulations are based on three fundamental governing equations of fluid dynamics: continuity, momentum, and energy equations. They concern the following physics laws:

1. Mass is conserved;
2. Newton’s second law;
3. Energy is conserved.

As mentioned above, there are two main simulation approaches for fluids, Eulerian and Lagrangian, although their flow-governing equations are equivalent [Wag87].

B.2.1 Lagrangian Method

The Lagrangian approach has been widely used in animation of natural phenomena. The works due to [Sta99], [PTB+03], [MMS04], [AH04] are only examples of the application of this approach to water and smoke. This approach treats the continuum like a particle system. Each point of the fluid or solid is labeled as a particle with a position $\vec{x}$ and a velocity $\vec{u}$. Considering the Lagrangian viewpoint, the incompressible Navier-Stokes equations are derived as a set of partial differential equations that hold throughout the fluid. They are written as

$$\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} + \frac{1}{\rho}\nabla p = \vec{g} + \nu \nabla \cdot \nabla \vec{u} \tag{B.16}$$

$$\nabla \cdot \vec{u} = 0. \tag{B.17}$$

The left hand side of the momentum equation (B.16) can be interpreted as the acceleration of a particle, while the right hand side is the net force exerted on such particle.

B.2.2 Eulerian Method

The Eulerian approach follows a different strategy, which is more commonly found in the simulation of liquids [FOK05, SU94, RNGF03]. Instead of tracking each particle, it looks at fixed points in space and sees how the fluid quantities measured at those points change with time. Thus, the fluid region is modeled as vector fields of fluids quantities; for example, if one considers a specific time and a given position, the vector field $\vec{v}(x, y, z, t)$ characterises the velocities, while the vector field $\vec{p}(x, y, z, t)$ measures the pressure inside the fluid. The incompressible Navier-Stokes equations have the following form in the Eulerian approach:

$$\frac{\partial \vec{u}}{\partial t} = -\vec{u} \cdot \nabla \vec{u} + \nu \nabla \cdot \nabla \vec{u} - \frac{1}{\rho} \nabla p + \vec{g} \tag{B.18}$$

$$\nabla \cdot \vec{u} = 0. \tag{B.19}$$

B.3 Equations of Cloud Motion

The process of cloud formation based on the parcel theory presented in Chapter 3 seems quite intuitive, but it is governed by the laws of thermodynamics. Understanding these laws helps in predicting the formation of clouds, as well as in making clear how the vertical profile of temperature in the atmosphere is determined.

Before condensation, air has no liquid water, i.e., we have dry air. A dry air parcel is described by three properties: temperature (T, in K), pressure (P, in N/m^2) and density (ρ, in kg/m^2). These properties are related by the ideal gas law [CBK11], which states that

$$P = \rho R T \tag{B.20}$$

where R is the gas constant that takes the value $R_d = 287.05307\,\text{Jkg}^{-1}\text{K}^{-1}$ for dry air, and $R_m = 461.5\,\text{Jkg}^{-1}\text{K}^{-1}$ for moist air. In practice, we use T_v (i.e., the virtual temperature) instead T in Equation (B.20), which is given approximately by

$$T_v = T(1 + 0.6\,d) \tag{B.21}$$

being d the water vapor density. The virtual temperature T_v is defined as the dry air temperature under the assumption that its pressure and density are the same as those of moist air. As the air parcel expands in response to a decrease in pressure, its

temperature also decreases. This process is called *adiabatic expansion* and is governed by the first law of thermodynamics [CBK11].

In the vertical direction, gravity is the most important external force acting on the atmosphere. The balance between pressure and gravity is known as the hydrostatic balance and environment pressure P_e and parcel pressure are said to be in hydrostatic balance when

$$\left(\frac{dP_e}{dz}\right)_z \approx -\rho_e(z)g \approx -\rho_p(z)g, \tag{B.22}$$

where g is the acceleration due to gravity ($g = 9.8m/s^2$), ρ_e and ρ_p stand for the environment and parcel densities, respectively. For the air parcel at $z + \Delta z$, the upward force is just the environmental pressure gradient, $(dP_e/dz)_{z+\Delta z}$ and the downward force is $-\rho_p(z + \Delta z)g$. From Newton's law of motion [Cro00],

$$\rho_p(z)\frac{dv}{dt} = -(dP_e/dz)_z - \rho_p(z)g \tag{B.23}$$

where v is the vertical velocity of the air parcel at z, and dv/dt denotes the vertical acceleration of the air parcel.

Equation (B.23) is the basic equation of motion for the air parcel at z, which expresses the vertical acceleration of the air parcel as the vertical gradient of the pressure perturbation and buoyancy. Once the parcel has been lifted to z, it has the same pressure as the environment, that is, $P_e(z) = P_p(z) = a$. Using the ideal gas law (Equation (B.20)), we have $\rho_e(z) = (a/R)[1/T_e(z)]$ and $\rho_p(z) = (a/R)[1/T_p(z)]$. Considering this and substituting (B.22) in (B.23), we obtain:

$$\frac{dv}{dt} = g\Big[T_p(z) - T_e(z)\Big]/T_e(z) \tag{B.24}$$

The ideal gas law and the laws of thermodynamics can be used to derive Poisson's equation for adiabatic processes, which relates to the temperature and pressure of a gas under adiabatic changes [RY89]:

$$\left(\frac{T}{T_0}\right) = \left(\frac{P}{P_0}\right)^k, \tag{B.25}$$

where T_0 and P_0 are the initial values of temperature and pressure, and T and P are the values after an adiabatic change in altitude and

$$k = \frac{R_d}{c_p} = \frac{c_p - c_v}{c_p} \approx 0.286, \tag{B.26}$$

where c_p and c_v are the specific heat capacities of dry air at constant pressure and volume, respectively.

In atmospheric sciences, potential temperature is a more convenient variable to account for adiabatic changes of temperature and pressure, since its value is constant under

adiabatic change of altitude. The potential temperature (θ) of a parcel can be defined as the temperature that the air parcel would have if it were moved adiabatically from pressure P and temperature T to pressure P_0. It is given by:

$$\theta = T\Big(\frac{P_0}{P}\Big)^{R_d/c_p}, \tag{B.27}$$

By using potential temperature instead of absolute temperature in Equation (B.24), air parcels of different altitudes can be compared regardless of the corresponding air pressure, so we have the final form of the equation of motion in terms of potential temperature:

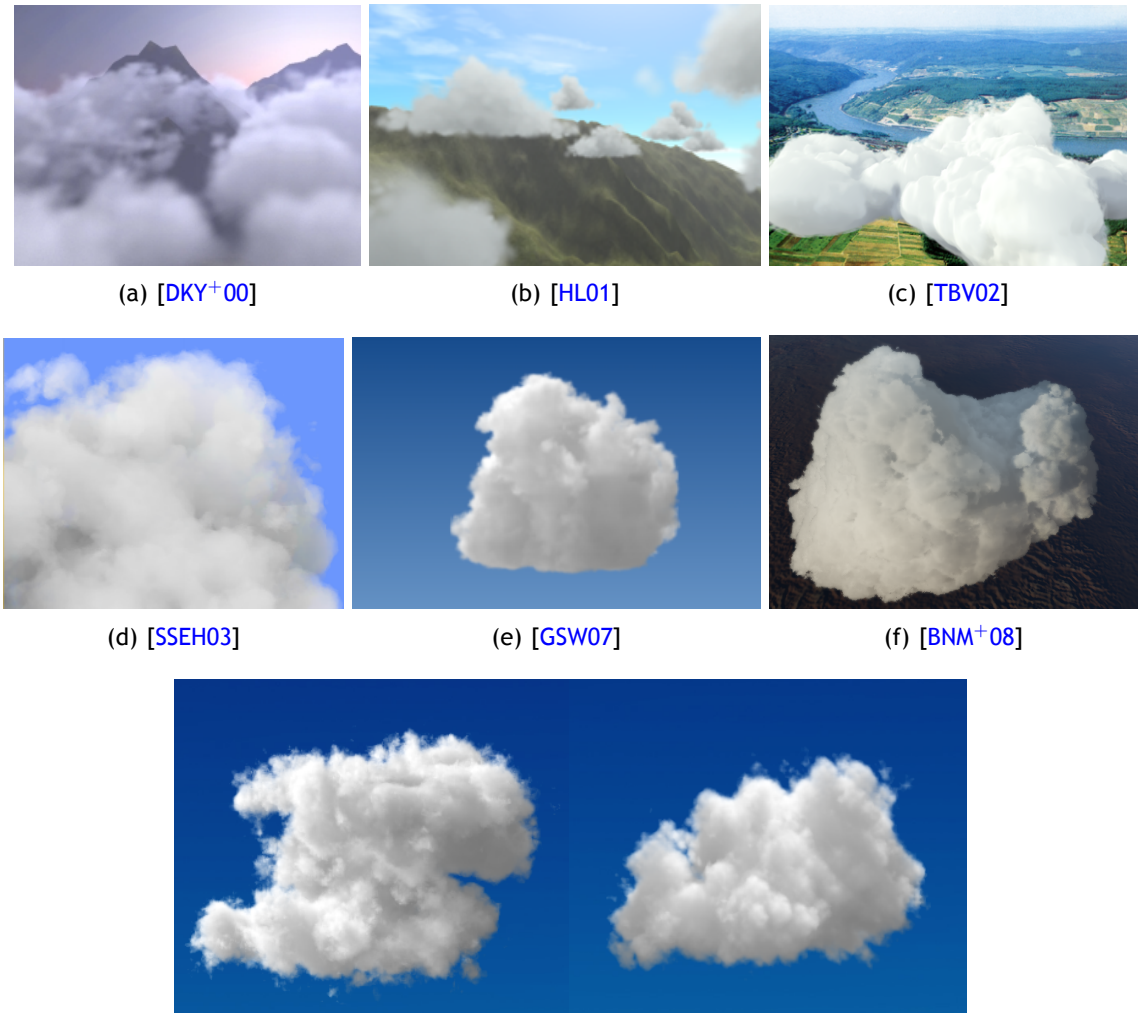
$$\frac{dv}{dt} = g\Big[\theta_p(z) - \theta_0(z)\Big]/\theta_0(z) \tag{B.28}$$

Therefore, Equation (B.28) can be used to determine the vertical displacement of an air parcel in the atmosphere by determining the potential temperature of both air parcel and ambient temperature, at a given pressure level.

Appendix C

Visual Comparison with Other Methods

In Figure C.1, we present several images of state-of-the-art clouds produced by several algorithms, including ours, for visual comparison sake. After a brief glance at these synthetic clouds, we intuitively observe that the clouds generated by our method are comparable to the state-of-the-art clouds, not to say they are more realistic than those. Nevertheless, those generated by the methods due to Schpok et al. [SSEH03] and Geist et al. [GSW07] seem to be very realistic as well.



(a) [DKY+00] (b) [HL01] (c) [TBV02]

(d) [SSEH03] (e) [GSW07] (f) [BNM+08]

(g) Our work

Figure C.1: Visual comparison between clouds generated by different methods.

Appendix D

2D User Interface

The 2D SkewT/LogP diagrammatic interface supports the core of our work. Here, all background curves are used to determine the parameters related to cloud formation. Moreover, this 2D interface also allows us to forecast weather, but this is out of the scope of this thesis. Figure D.1 shows the 2D SkewT/LogP interface for illustration purposes, while Figures D.2 and D.3 plot two soundings extracted from http://www.twisterdata.com.

The numbered items in Figure D.1 refer to a number of data, namely:

① Sounding parameters and cloud levels:

- *Ambient temperature* and *dewpoint temperature* refer to the sounding temperature at the P_0 pressure level, i.e., temperatures at ground level, extracted from sounding data;
- *LCL*, *CCL*, *LFC* and *EL* are the standard levels of the atmosphere related to cloud levels. They are determined from the sounding data using the geometric procedures described in Chapters 3 and 4.

② *Stability indices*. These indices relate to thunderstorm forecasting. The calculation of these indices is not described in the current thesis, since thunderstorm prediction is out of its scope.

③ *Wind profile*. Here, the wind profile is represented in a hodograph. As aforementioned, a hodograph allows meteorologists to plot the wind field from soundings to determine the wind shear, which has a great impact on thunderstorm formation. In our work, we are more interested in a vertical profile of the wind data, and the relation between the hodograph and the vertical wind profile is presented in Section 3.4.

④ This plot represents the variation of the relative humidity in the vertical extent of the atmosphere. Usually, at higher levels in the atmosphere (e.g., 100 mb), there is almost no moisture in the air.

⑤ *Background curves of a SkewT/LogP diagram*. Our data structure includes a full implementation of these curves, since they are very important in the determination of the cloud levels referred in ①. Also, we are able to determine the intersections between the sounding curves and these background curves in order to "feed" the motion equation, as particles rise in the atmosphere.

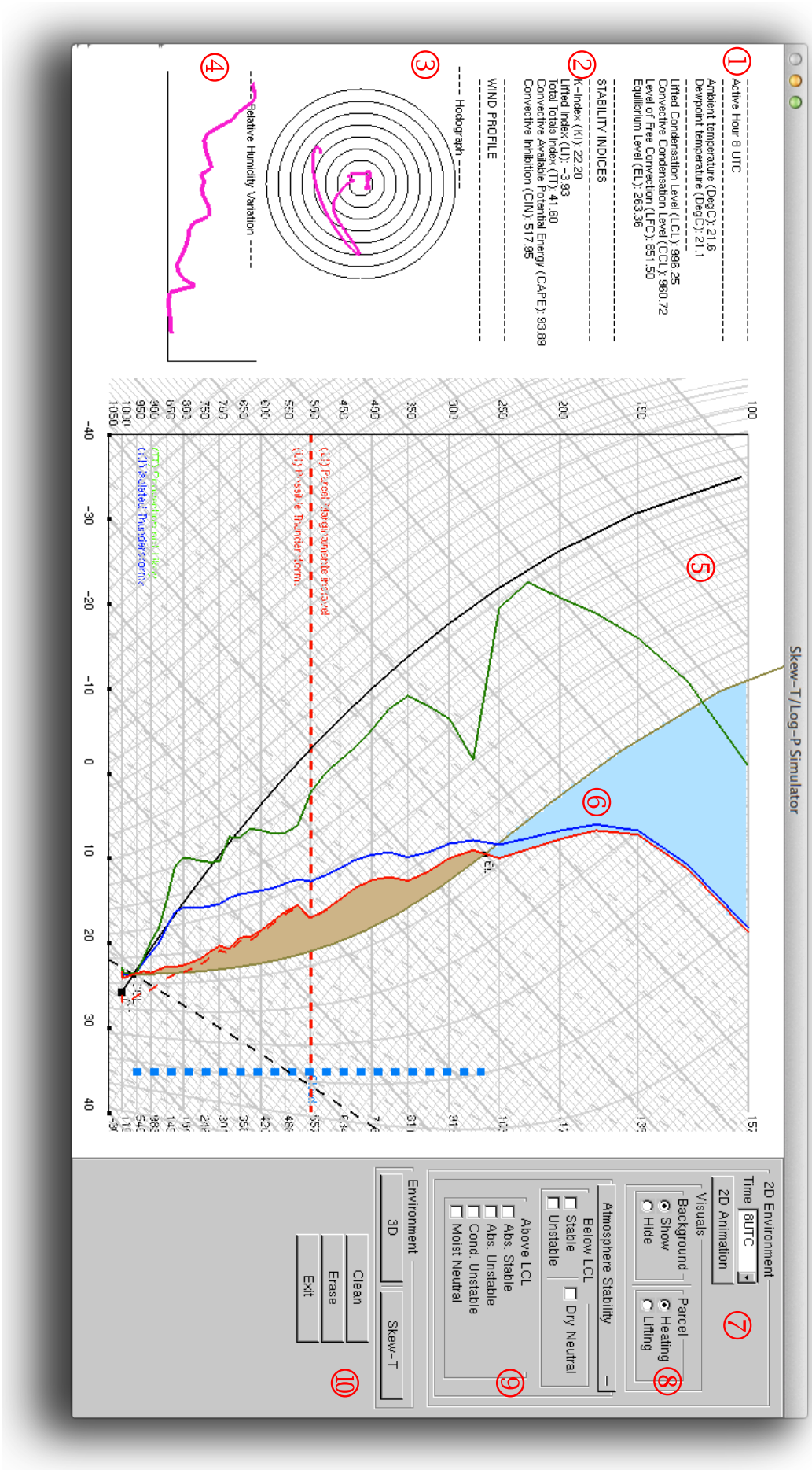


Figure D.1: Our 2D interface that shows the implementation of all background curves, sounding data and atmospheric parameters (some related to cloud formation and other related to weather forecast).

⑥ *Sounding curves*. Several curves are represented in this diagram: the ambient temperature (in red); the dew-point temperature (in green); and the potential temperature curve (in blue). In addition, three auxiliary curves for parameters calculation are plotted: dry-adiabat curve and the mixing ratio curve used to estimate the CCL, and the moist adiabat curve to estimate the EL. Also, the light brown and light blue areas represent the convective inhibition (CIN) and convective available potential energy (CAPE), which correspond to positive and negative energy areas for cloud motion.

⑦ *Browsing sounding data*. Allows us to load a specific sounding or, alternatively, to browse atmospheric soundings loaded in our system.

⑧ *Visuals*. Allows us to show or hide the background curves, and chose the lifting method: heating (related to convection) and lifting.

⑨ *Atmosphere stability*. Determines the vertical stability and instability of the atmosphere, and the levels where they occur. This is an important factor to predict the existence of cloud layers, given that turbulence may occur between such layers. However, more information about the behaviour of the atmosphere is required to generate those layers, which is out of scope of this thesis.

⑩ *Interface options*. These buttons allow us to swap between the 3D simulator and the 2D SkewT/LogP diagram builder. In the 3D simulator, clouds are generated and the SkewT/LogP (diagram builder) button allows us to edit the sounding curves and cloud formation levels.

As aforementioned, we use data from weather agencies in tabular format. We read this data in our system, and for each hour (UTC) we obtain a complete profile of the atmosphere during the day. In Figures D.2 and D.3, we present two soundings in the 2D interface.

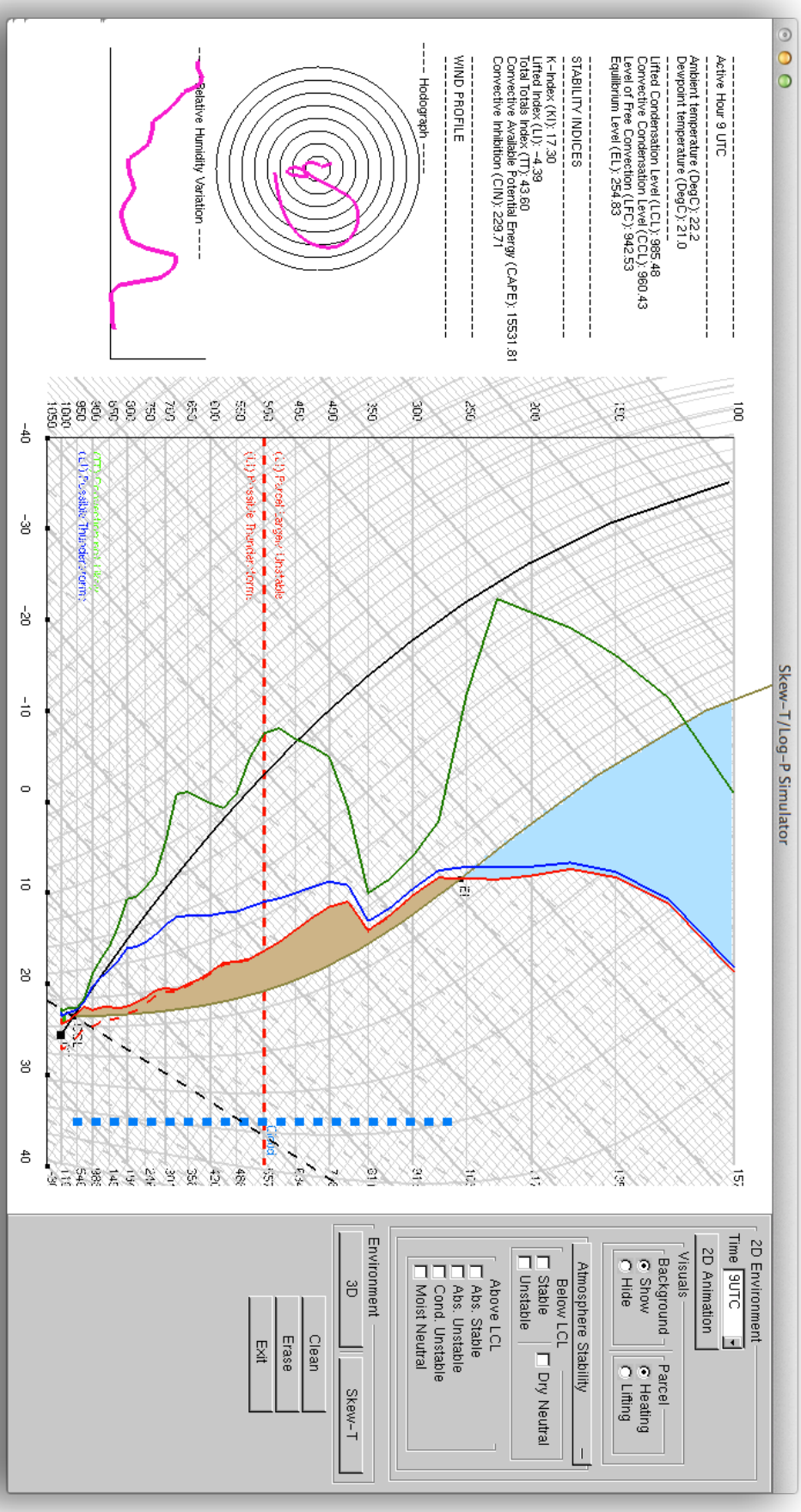


Figure D.2: Sounding loaded into our system at 9UTC.

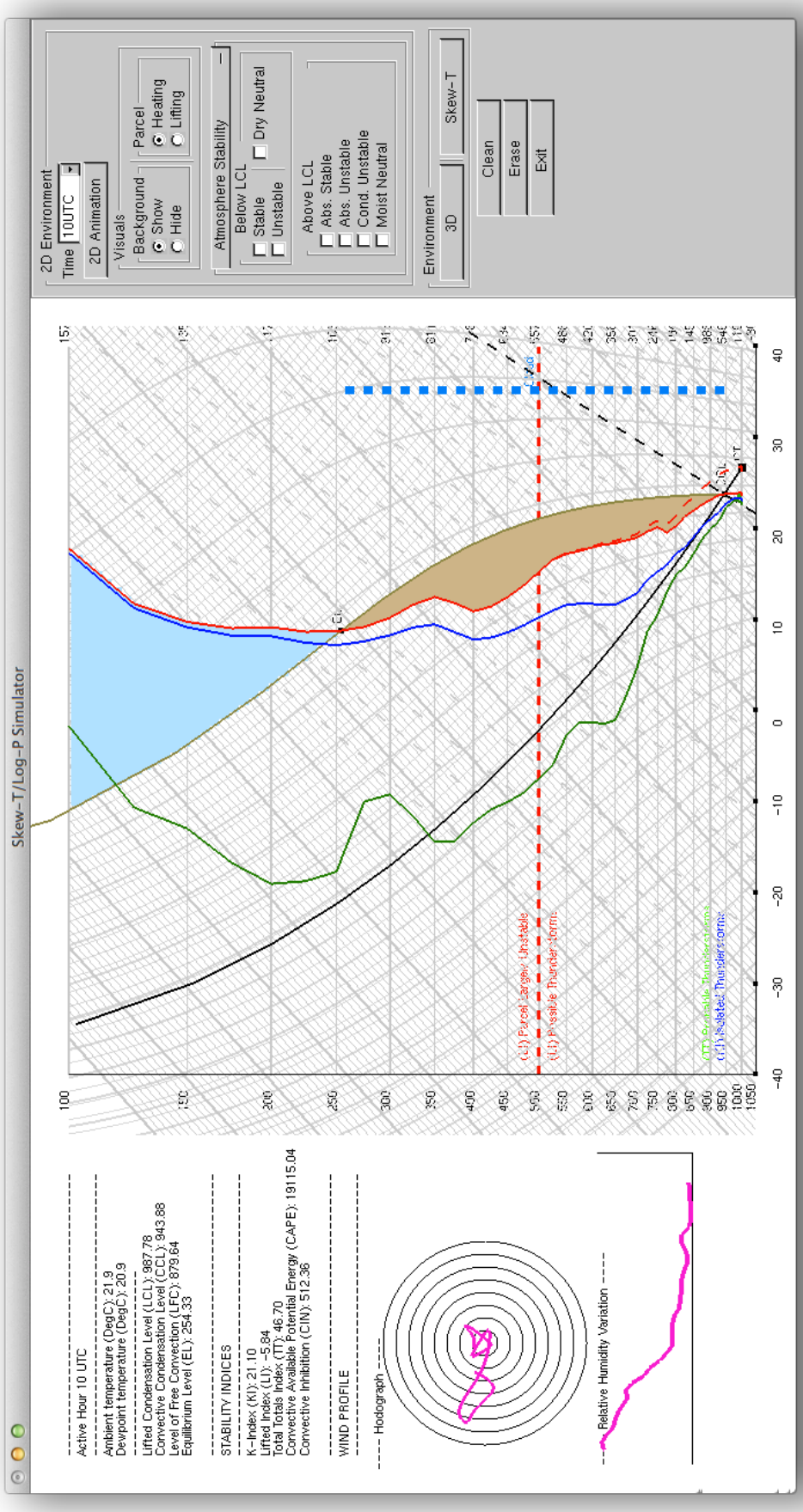


Figure D.3: Sounding loaded into our system at 10UTC.

Glossary

Adiabatic process	A process in which a system does not interact with its surroundings by virtue of a temperature difference between them. In an adiabatic process any change in internal energy (for a system of fixed mass) is solely a consequence of working. For an ideal gas, and for most atmospheric systems, compression results in warming, expansion results in cooling.
Advection	The process of transport associated to an atmospheric property by the mass motion (velocity field) of the atmosphere; also, the rate of change of the value of the advected property at a given point.
Air mass	A widespread body of air, whose properties can be identified as 1) having been established while that air was situated over a particular region of Earth's surface (air mass source region), and 2) undergoing specific modifications while in transit away from the source region.
Atmospheric pressure	the pressure exerted by the atmosphere as a consequence of gravitational attraction exerted upon the "column" of air lying directly above the point in question.
Buoyancy	The upward force exerted upon a parcel of fluid (or an object within the fluid) in a gravitational field by virtue of the density difference between the parcel (or object) and that of the surrounding fluid. It is also called Archimedean buoyant force.
Condensation	The process of changing from a gaseous to a liquid or solid state.
Convection	Motion that is predominantly vertical and driven by buoyancy forces arising from static instability, with locally significant deviations from hydrostatic equilibrium.
Coriolis force	A force relative to Earth's surface that causes deflection of moving objects to the right in the northern hemisphere and to the left in the southern hemisphere due to Earth's rotation.
Dewpoint	(or dewpoint temperature.) The temperature to which a given air parcel must be cooled at constant pressure and constant water vapor content in order to make saturation occur.

Dry adiabats	A line of constant potential temperature on a thermodynamic diagram that describes the dry air temperature profile of air parcels in the atmosphere.
Isopleth	A line drawn on a thermodynamic diagram through all points having the same value of some measurable quantity.
Isotherms	A line of equal or constant temperature. A distinction is made, infrequently, between a line representing equal temperature in space, choroisotherm, and one representing constant temperature in time, chronoisotherm.
Leeward	The downwind side of the mountain, ridge, or other flow obstacle away from the large-scale or ridge-top flow direction.
Moist adiabats	On a thermodynamic diagram, a line representing a moist adiabatic expansion of an air parcel. They are also known as pseudoadiabats.
Orographic lifting	Ascending air flow caused by mountains. Even though this term strictly refers to lifting by mountain presence, it is sometimes extended to include effects of hills or long sloping topography.
Phase transition	A thermodynamic process in which a substance changes from one phase to another.
Thermodynamics	The branch of physics concerned with the conversion of different forms of energy.Thermodynamics introduces a new concept, called temperature, which is absent from classical mechanics and other branches of physics.
Thermodynamic diagram	diagrams used to represent the thermodynamic states of a material (typically fluid) and the consequences of manipulating this material.
Troposphere	The portion of the atmosphere from Earth's surface to the tropopause; that is, the lowest 10 - 20 km of the atmosphere; the portion of the atmosphere where most weather phenomena occurs.
Vapour density	In a system of moist air, the ratio of the mass of water vapour present to the volume occupied by the mixture; that is, the density of the water vapour component.

Vapour pressure	The pressure exerted by a vapour. In meteorology, vapour pressure is used almost exclusively to denote the partial pressure of water vapour in the atmosphere.
Water vapour	Water substance in vapour form; one of the most important constituents of the atmosphere. Also called aqueous vapour or moisture.
Weather forecast	An assessment of the future state of the atmosphere with respect to precipitation, clouds, winds, and temperature, often using numerical simulations.
Windward	The upwind side of the mountain.