Fachbereich 4: Informatik

Interactive Simulation of Clouds Based on Fluid Dynamics

Diplomarbeit

zur Erlangung des Grades eines Diplom-Informatikers
im Studiengang Computervisualistik

vorgelegt von

Oliver Koehler

Erstgutachter: Prof. Dr.-Ing. Stefan Müller
(Institut für Computervisualistik, AG Computergraphik)

Zweitgutachter:

Koblenz, im Juni 2009

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

	Ja	Nein
Mit der Einstellung der Arbeit in die Bibliothek bin ich einverstanden.	☐	☐
Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.	☐	☐

.....
(Ort, Datum) (Unterschrift)

Contents

List of Figures

1 Introduction

Humankind has always been fascinated by clouds. It seems amazing how these enormous and seemingly solid structures just float around in the sky and how they appear and disappear out of nothing. The work of countless poets, authors and songwriters has been influenced by the observation of clouds and the phenomena associated with them. From the menacing atmosphere transmitted by an approaching thunderstorm to the colorful and soothing display of clouds at sunset, they can evoke a wide variety of emotions. Painters of all styles and epochs have tried to capture these spectacular scenes in their paintings.

Before today's scientific weather forecast came to be, looking at the skies was the only possible way to get an indication of how the weather might be in the near future. In pre-industrialized, rural society, the ability to recognize and interpret cloud patterns was of vital importance. Today, even though the understanding about cloud formations is not necessary for survival anymore, it can still be useful to complement the official weather forecast with local details.

In the computer science community, and especially in computer graphics, a lot of research has been done in order to achieve realistic renditions of clouds, because they are an indispensable element of any outdoor scene. However, most cloud models developed did not simulate the actual physical processes but took a more artistic approach, since a physically motivated simulation of clouds is computationally very expensive. Only recently has sufficient computational power become available to use physics-based cloud simulation in computer graphics. Still, certain assumptions have to be made and some concepts have to be simplified in order to keep frame rates at interactive levels.

The cloud model developed in this thesis attempts to correctly model the physical processes involved in cloud formation. It incorporates concepts from fluid dynamics and thermodynamics, and renders the results to the screen using techniques from computer graphics.

The goal is to achieve a simulation that runs at interactive frame rates and that allows the user to visually experience the basic processes involved in cloud formation. It should also be possible to interactively change important parameters of the simulation and observe the consequences in near real-time. It was therefore necessary to develop visual representations of different aspects of the cloud simulation, such as the velocity or temperature.

This thesis is organized as follows: Chapter 2 introduces the most important works and approaches for the modeling and rendering of clouds. Both procedural and physics based approaches are presented.

In chapter 3, the most important quantities and mechanisms involved in cloud formation in the atmosphere are presented in a comprehensible

manner, while chapter 4 puts these principles on a more formal basis. It describes the models and equations used in the cloud model, based on atmospheric physics, and also explains where and why certain aspects were simplified. Chapter 5 then describes how the aforementioned equations and concepts are solved on the computer. It also sheds some light on the rendering method that has been employed, as well as explaining the visual helps that were developed for the visualization of the cloud dynamics. The results are presented in chapter 6, also discussing the limitations and shortcomings of the proposed cloud model and simulation method.

2 Related Works

Clouds are omnipresent in any outdoor scene and it therefore comes as no surprise that a lot of research has been done with the aim of creating realistically looking clouds. Many of these techniques are not intended for a physically accurate simulation but rather focus on the visual aspects of clouds. However, in order to achieve an exhaustive description of previous works related to the problem at hand, the following section will provide an overview over most of the techniques that have been used for the simulation and rendering of clouds.

When simulating clouds, two distinct fields of research have to be taken into account. First, there is the simulation of the physical processes that govern the formation of clouds. These processes can be described by the laws of atmospheric fluid dynamics. While these laws have been known in meteorology for quite some time, only lately (e.g. Since the end of the 1990s) do consumer class workstations dispose of sufficient computational power to perform the required calculations at interactive rates. Especially the advent of programmable graphics processors (GPUs) has fueled this development, since they are designed for parallel execution of several calculations at once.

The second important aspect of cloud simulation is rendering. A cloud dynamics simulation typically delivers a three-dimensional density distribution, and therefore requires a method to render this volumetric data.

2.1 Cloud Modeling

The modeling of clouds is a challenge for every developer due to their amorphous nature. Since they constantly appear, disappear, and change shape, clouds cannot easily be modeled using solid geometric primitives, although such approaches do exist. The distinction that is taken here in order to classify different cloud modeling techniques differentiates between procedural approaches and physically motivated techniques. While the first tackle the problem more from an artist's perspective and focus on visually convincing cloud modeling and rendering techniques, physically motivated approaches try to capture and simulate the underlying physical processes that are responsible for the formation of clouds. It should however be noted, that the different techniques presented here are by no means mutually exclusive. In fact, most works use a combination of several methods.

2.1.1 Procedural Techniques

Throughout the history of computer graphics, procedural approaches have been very popular for the modeling of natural phenomena such as clouds,

fire, smoke, or for the creation of textures that are intended to resemble natural materials like marble, wood, or stone. Procedural techniques use algorithms to specify the properties of the desired materials (for example its color or opacity value), and therefore don't rely on photographs or handmade images as input. This reduces memory requirements and eliminates the need to have an artist manually design these models and textures. Another advantage of procedural techniques is that they usually specify a number of parameters (such as a value for the 'roughness' of a material) that offer an increased level of flexibility to the animator/modeler [3]. In fact, most professional animating software suites (like Maya or 3D Studio Max) offer some procedural methods to model natural phenomena like smoke or fire.

Noise One procedural technique that has been used in several works in cloud modeling is procedural noise. Noise is an adequate method for modeling seemingly "chaotic" phenomena like clouds or smoke, and it is also useful to represent the irregularities that many natural materials like stone or wood typically exhibit. It can also be employed to give materials a more "rugged" or "used" look.

In 1989, Ken Perlin [17] presented a method to create continuous random volumetric data based on procedural noise, which is applicable not only to clouds, but to a wide array of materials. His methods extends the idea of procedural solid textures to three dimensions. Clouds are modeled as density fields that are then modulated by the sum of a series of pseudo-random noise functions at different amplitudes and frequencies [17]. Figure 1 shows a sphere that has been contorted with a fractal sum of noise functions.

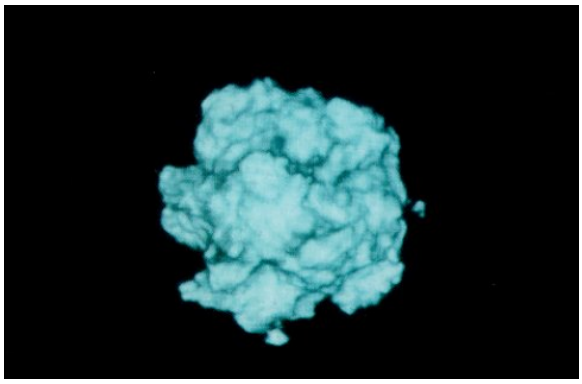

Figure 1: Fractal Sphere created with a noise function, image courtesy of Perlin [17]

In 2004 Ebert presented an approach to modeling clouds that also relies on noise and turbulence [3]. He models the cloud macro structure as

a union of implicit geometric primitives like spheres, cylinders, or ellipsoids. These are placed by the user, who thus directly influences the basic shape of the clouds to be. Comparable to Perlin, the macro structure of the clouds is then perturbed by procedural noise and turbulence functions that account for the small-scale details, the cloud's micro structure. Depending on the primitives used for the cloud's macro structure and several parameters like thickness, falloff, or density, different cloud types can be created. Coupled with a physically based volume renderer that accounts for the effects of light scattering, very realistic images and animations of clouds can be achieved, however not in real time. Figure 2 shows two examples of different cloud types.

Figure 2: Cumulus (left) and cirrostratus clouds (right) rendered with Ebert's method, images courtesy of Ebert [3]

Building on Ebert's work and using the same two-level approach, Schpok et al. [19] extended the technique to draw on the additional computing power available through consumer graphics hardware. By reallocating computationally expensive tasks to the graphics processor and using a less refined illumination model, they were able to reach interactive frame rates, between 5 and 30 fps [19]. See figure 3 for examples.

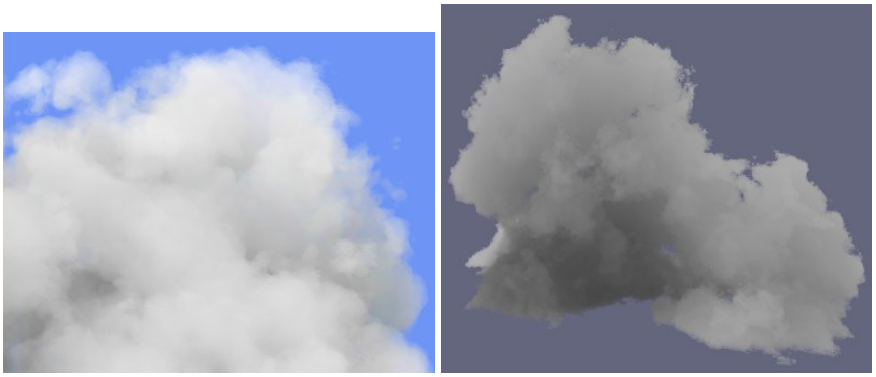

Figure 3: Cumulus clouds rendered with Schpok's GPU-assisted method, images courtesy of Schpok [19]

Textured Solids One significant disadvantage of volumetric cloud representations is their computational complexity. Especially in real-time applications such as flight simulators, fast modeling and rendering techniques are indispensable. There have been several attempts to model clouds as textured solids, because this representation promises the best performance.

One early work following this approach was presented by Geoffrey Gardner in 1985 [7]. His cloud model consists of three building blocks: A two-dimensional sky plane is used to represent layer clouds and serves as a backdrop. In this way, clouds that are further away are not explicitly modeled in 3D, thus increasing performance. The macro structure of cumulus clouds and other cloud types that cannot be adequately modeled in two dimensions because they are relatively close to the camera, consists of ellipsoids. In order to allow for more complex cloud shapes, several ellipsoids can be linked together. Similar to Ebert [3], noise textures are used to modulate the shading intensity and translucency. Given that his work was published almost 25 years ago, the results (see figure 4) are quite impressive, although it has to be noted that at the time, rendering of a single image took between 20 and 25 minutes [7].

Figure 4: A cloudscape combining 2D- and 3D-clouds by Gardner, image courtesy of Gardner [7]

Based on Gardner's work, Elinas and Stuerzlinger [4] extended this approach in 2000 to make use of new technological advances like photon maps using hardware-accelerated OpenGl. They also changed the way their textures are created: While Gardner relied on spectral synthesis, they used Perlin noise to create the textures that regulate transparency and color. Figure 5 shows a comparison between different noise textures.

Particle Systems Another cloud modeling technique is based on particles. Particles are small geometric primitives (often spheres) that can be described by their position in three-dimensional space and only a few other

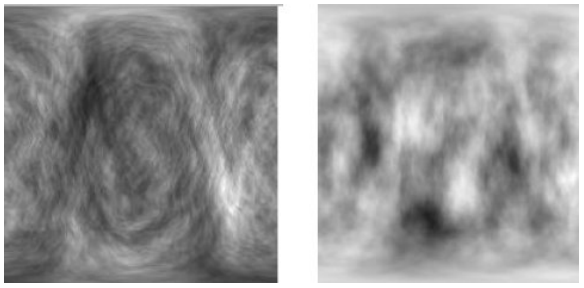

Figure 5: Noise textures as used by Gardner (left) and Elinas/Stuerzlinger (right), courtesy of Elinas/Stuerzlinger [4]

parameters (like radius, color, texture, etc.) [8]. As such, they are relatively easy to implement and inexpensive to manage, and are therefore often used in real-time applications. Many systems that model gaseous phenomena like smoke, fire or clouds rely on particle systems, since they can be used in conjunction with an underlying physics engine. However, the advantage of using particle systems diminishes somewhat when a lot of small-scale details are required, because the number of particles has to be increased considerably.

Lastra and Harris [13] presented their cloud modeling approach in 2001. They represent clouds as a collection of particles and render each particle as a small, textured sprite. Building on a technique introduced by Gardner [7], they introduced the use of impostors, 2D-textures that represent clouds that are farther away. Since only near clouds are modeled in 3D, they achieved very high frame rates. However, their method could only represent static clouds and did not involve any sort physical model.

Figure 6: Real-time static clouds created by Lastra et al., image taken from [13]

2.1.2 Physics-Based Cloud Simulation

The procedural approaches to cloud modeling presented so far are aimed mostly at achieving visually convincing representations of clouds, and are less concerned with the underlying physics. In fact, from an artist's point of view, a procedural approach is preferable because it offers more control over the shape and appearance of the clouds.

Physics-based cloud modeling takes a different approach. The goal here is to accurately portray the physical processes that are responsible for the formation of clouds in nature. The clouds that are modeled in this fashion should come into being as the result of applying the laws of nature and physics, combined with techniques from computer graphics to render the results. As such, an animator has only indirect control over the process, because he can typically only specify certain parameters that correspond to real physical quantities and then has to "let nature do the rest".

Fluid Dynamics Physics-based cloud dynamics are closely linked with fluid dynamics, because the laws from fluid dynamics (namely the Navier-Stokes equations) can be used to accurately describe the flow of air masses in the atmosphere. All approaches to modeling the dynamics of clouds implement some method to solve these equations. Since fluid dynamics and the Navier-Stokes equations play such an integral part in the simulation of clouds, some works that were important in the development of fluid solvers are presented first, although these works are generally not focused on clouds, but comparable phenomena such as smoke, fire, or liquids.

The Navier-Stokes equations have been known for more than a hundred years. They model fluid flow using a set of non-linear partial differential equations in terms of velocity and pressure (see chapter 4 for an in-depth explanation). In 1995, Foster and Metaxas [6] described a incremental method to numerically solve these equations. Their approach was based on a rectangular grid that divides space into voxels, and the Navier-Stokes equations were solved incrementally for each voxel. The problem with this approach was that it suffered from instability when the time step for the simulation became too large. Allowing only small time steps however, increases computation time.

Based on the work of Foster and Metaxas, Stam developed a method that guaranteed unconditional stability for any time step size [20][21]. Instead of the previous explicit time stepping scheme, he used an implicit, semi-Lagrangian method that borrowed ideas from particle systems. Today, many works in computer graphics that try to simulate some kind of fluid flow are based on Stam's method, and it has also been used in this thesis (see chapter 5).

The most important drawback of Stam's method is that it leads to dissipation, thus damping out some of the small-scale swirls and turbulences

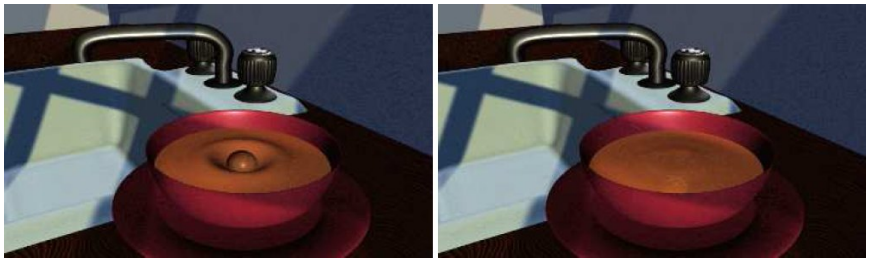

Figure 7: A screen shot from Foster & Metaxas work on fluid simulation, image courtesy of Foster & Metaxas [6]

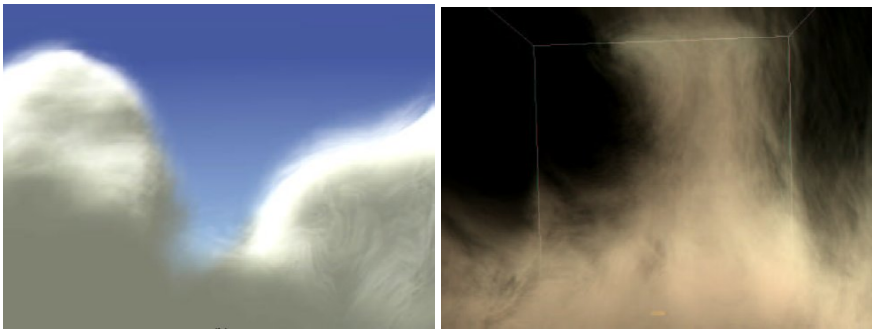

Figure 8: A cloud and smoke simulated with Stam's method, images courtesy of Stam [20]

typically encountered in fluid motion. Fedkiw et al. approached this problem in [5], introducing a method called "vorticity confinement" that reintroduces lost rotational energy back into the fluid. This method allows more detailed fluid flows.

Cloud Dynamics The first physics based models of cloud dynamics were developed by meteorologists and atmospheric scientists. As they were used only as scientific tools, they were focused on accurately modeling the physical processes, and less on performance and rendering. The enormous computational complexity involved in simulating cloud dynamics further hindered the application of physically motivated cloud models to any other domain but meteorology and atmospheric sciences.

The first in developing a physically based cloud model for computer graphics were Kajiya and von Herzen in 1984 [12]. They solved the Navier-Stokes equations of incompressible fluid flow and also incorporated a thermodynamics model and a simple water continuity model [16]. With their method they achieved solving one step of cloud evolution in about 10 seconds on a 10x10x20 grid.

A similar method was presented by Overby in 2002 [16]. In this work, the stable fluid simulation algorithm developed by Stam [20] was used to

Figure 9: A picture from Fedkiw's smoke simulation that preserves small scale details, images courtesy of Fedkiw et al. [5]

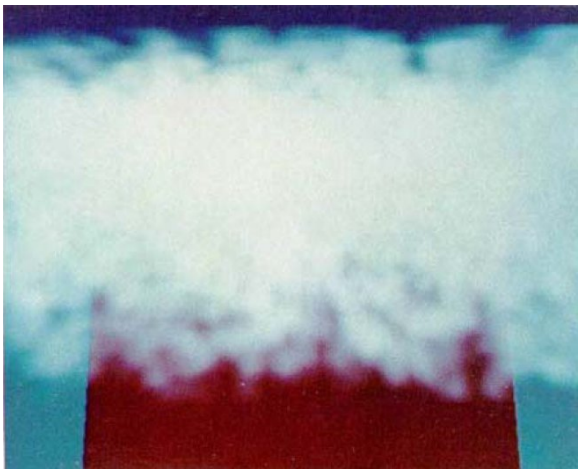

Figure 10: Image computed with Kajiya's method, courtesy of Kajiya/von Herzen [12]

solve the Navier-Stokes equations. Overby modeled the water in the system in terms of relative humidity, with saturation values directly proportional to pressure. The model also accounted for the existence of cloud condensation nuclei, small particles in the air that have can lower or raise the saturation threshold. The user interface allowed to create different cloud types (cumulus, stratus) by changing the initial conditions of the simulation.

The volumetric data resulting from the simulation was then rendered using alpha-blended billboards, thus achieving a performance of one iteration per second on a Pentium III with a 15x50x15 grid [16].

The work presented by Mark Harris in 2003 [8] was the first that achieved frame rates fit for real-time applications by consequently exploiting the

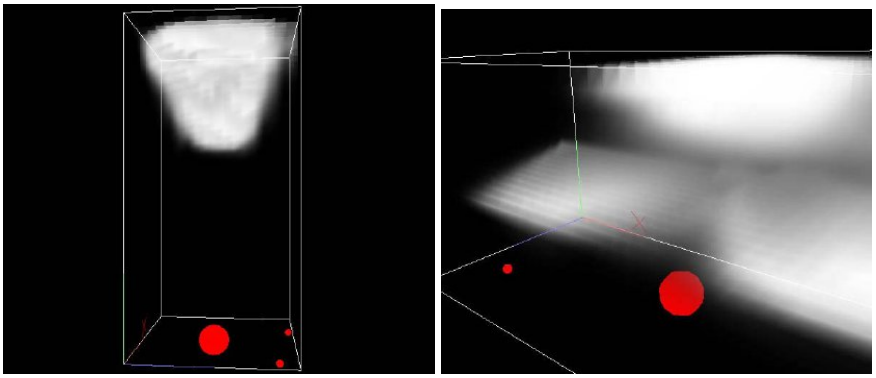

Figure 11: Cumulus (left) and stratus clouds (right) generated with Overby's method. The red dots in the left image represent heat and vapor sources. Image courtesy of Overby [16].

computational power of graphics hardware. He developed a fairly complicated cloud model and used Stam's stable fluid algorithm [20] for the dynamics, and implemented both on the GPU. In order to further enhance performance, a technique called "simulation amortization" was developed that spreads the dynamics calculations over several frames.

For the rendering of the clouds, algorithms were developed that simulate multiple forward light scattering, and that are applicable both to particle and voxel based cloud representations. In order to speed up the rendering process, dynamically generated impostors were used. These are basically two-dimensional textures that can be applied to represent clouds that are so far away from the camera that they do not have to be represented in 3D. Similar techniques had already been described in [7], [22], and [13].

Harris cloud simulation achieved frame rates of up to 3.9 frames per second on a 64x64x64 grid and using GeForce 5900 Ultra in conjunction with a Pentium IV CPU.

Figure 12: Static, particle-based clouds rendered with Harris' method, courtesy of Harris [8]

Other approaches to modeling the dynamics of clouds have experi-

mented with simpler (and therefore faster) approximations of fluid flow.

in 2000, Dobashi et al. [2] presented a method that approximates the behavior of clouds with cellular automatons. This method models cloud evolution on a voxel grid, just like the other physics-based approaches. The state values however are represented as logical (binary) values that change according to logical transition rules that simulate evaporation, condensation, etc. While this method is very fast, it is not able to simulate the processes of cloud formation accurately (see figure 13).

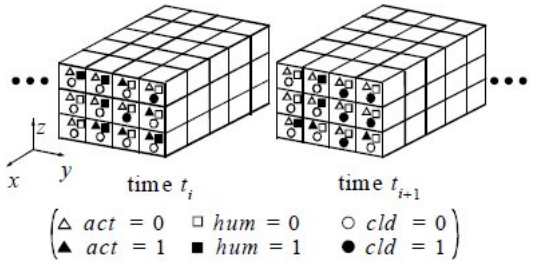

Figure 13: Binary values represent the state values in Dobashi's method based on cellular automatons. Image taken from [2]

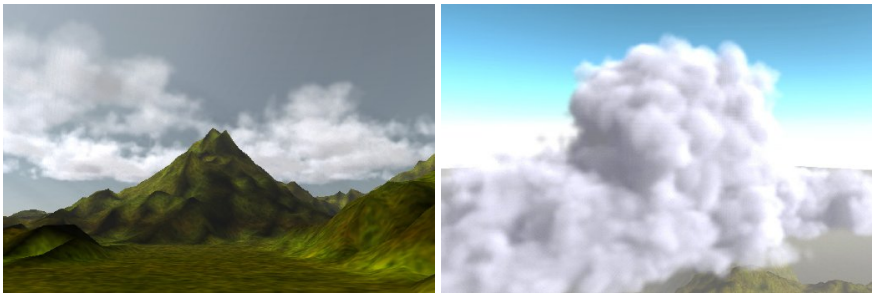

Figure 14: Clouds created by Dobashi et al., images taken from [15]

An extension of this method was presented by Myiazaki et al. in 2001 [15]. The basic idea was to use a concept called "coupled map lattices" (CML) instead of the cellular automatons, because they allow to use continuous instead of only binary values. The transition rules were than replaced by an approximation of the incompressible Navier-Stokes equations. The model presented in 2001 [15] was able to depict different cloud types, while another work presented in 2002 [14] explored more accurate ways to model cumulus clouds. Both these models included a water continuity model and also accounted for latent heat release due to phase changes.

All these works used volume rendering techniques using metaballs and splatting. Their rendering also included a lighting model and could depict shafts of light in between clouds. However, this rendering technique, al-

though producing very convincing images, is not suited for real-time applications.

2.2 Rendering Techniques for Volumetric Data

Virtually all works that realize a simulation of clouds based on the laws of atmospheric physics (or an approximation of them) model clouds on a voxel grid. Using this technique, the results of the simulation are provided in the form of a three-dimensional distribution of state values. Therefore they all require some volume rendering technique to visualize the results.

One possible method to render this data is ray tracing. However, since the goal of this thesis is to achieve interactive frame rates, ray tracing is not an option because it is too computationally expensive. The same is true for the metaball splatting technique developed by Dobashi et al. [2].

The method used in this thesis renders alpha-blended slice from front to back.

2.3 Summary

There is a multitude of different approaches available to render and simulate clouds. However, the selection of works presented here also reflects the eternal dilemma of computer graphics, to choose between quality and performance. Since the stated goal of this thesis is to achieve interactive frame rates while as accurately as possible simulating the physics of cloud formation, only few works could be used as orientation.

The cloud model developed in this thesis is most similar to those presented by Overby [16] and Harris [8]. Both model the physical processes involved in cloud dynamics. While this thesis uses the more accurate dynamics model presented by Harris, it was also influenced by the idea of simulating different cloud types introduced by Overby. Since the focus in this work lies more on depicting and visualizing the physical processes instead of achieving visually realistic results, a much simpler cloud rendering method was chosen for the sake of performance (presented in chapter 5.2).

3 Cloud Formation in the Atmosphere

The following chapter serves as an introduction to the formation of clouds. It focuses on the description of the most important concepts involved in the formation of clouds, and it is designed to be as comprehensible as possible. The ideas presented here will then be put on a more theoretical foundation in chapter 4.

3.1 Air Parcels

In order to better understand the processes involved in cloud formation, it is common in meteorology to study the behavior of air parcels. Such an air parcel is a conceptual tool that can be imagined as a thermically isolated volume of air that is able to expand and contract without any exterior force, and that can be traced relative to its surroundings. Although this idea might at first seem unrealistic, it actually matches the behavior real volumes of air quite well[9], because air has very low thermal conductivity.

3.2 Water and Humidity

Water is the most important property in climatology and meteorology. It occurs in all three physical states: as gas in the form of vapor, as a liquid in the form of water and water droplets, and in its solid state in the form of ice and snow. The weather on our planet and especially the formation of clouds rely heavily on the state changes that water undergoes in the atmosphere due to changing temperature and/or pressure. On a global scale, it is water and air currents that are responsible for the climatic conditions on our planet. For example, the gulf stream, a warm water current that originates in the gulf of Mexico and extends all the way to western Europe, is responsible for the moderate climate we enjoy in central and northern Europe.

The air on our planet always contains some amount of water vapor that comes into being through the process of evaporation. The majority of evaporation occurs over the oceans and other open waters such as lakes and rivers, but a certain part is also due to transpiration of plants, humans, and animals, as well as evaporation of ground water. It is estimated that the total amount of water present on earth is about 1,4 billion km. Of this total amount, less than 0,001%, or 13000 km is contained in the atmosphere as humidity and clouds [9]. Although it is but a tiny fracture, this atmospheric water is completely responsible for the formation of the clouds all over the world. It is generally referred to as humidity. There are several commonly used quantities that describe the relationship between air and vapor:

Absolute Humidity It describes the absolute amount of water vapor that is contained in a specific volume of air. It is usually specified in grams per cubic meter, but different units, such as pounds per cubic foot, are also commonly used in different parts of the world. The major inconvenience with this quantity is that it it is only constant for a certain altitude. When an air parcel changes its altitude, its volume also changes due to different pressure conditions. As a consequence, the absolute humidity changes as well.

Specific Humidity This quantity specifies the humidity in grams of water vapor per kg of air. By using mass instead of volume, specific humidity stays constant regardless of altitude and pressure. It is therefore much more commonly used. In the cloud model developed for this thesis, humidity is specified in this fashion. It is also often referred to as "water vapor mixing ratio".

Relative Humidity The maximum amount of water vapor that air can contain depends directly on its temperature. Warm air can contain much more vapor than cold air. Relative humidity is calculated as the quotient between the specific humidity and the maximum amount of vapor possible, and it is usually expressed as a percentage. In other words, it expresses to what percentage the air is saturated with vapor. This quantity is very useful in weather forecast, as a high relative humidity increases the likelihood of precipitation. On the other hand, it can be difficult to work with due to its direct dependence on temperature. An indication of temperature is therefore required in order to give the relative humidity any meaning.

Dew Point Another quantity that is closely related to humidity and that is needed in order to understand and calculate the relative humidity is the dew point. As mentioned above, the maximum amount of vapor that can be contained in the air depends on its temperature. When an air parcel with a given specific humidity cools down, its relative humidity rises. Eventually, the relative humidity will reach 100% and the air is thus fully saturated with vapor. This point is called the dew point. Figure 15 shows the so-called dew point curve (at sea level) that indicates how many grams of water vapor can be contained in a kg of air at a given temperature.

Given these quantities it is now possible to describe the basic physical process, that leads to the formation of clouds. Once a parcel of air passes 100% relative humidity/its dew point/its saturation point, some of the vapor contained in it has to be expelled. At this point, condensation sets in. A certain part of the water in the air changes its physical state from gaseous to liquid and tiny water droplets begin to form. These droplets then become visible as clouds and fog. The exact amount of vapor that changes its

physical state to water is such that the surrounding air always stays 100% saturated. The question then is, what circumstances are responsible for air to become saturated with vapor and therefore form clouds? The answer to this question can be obtained by again looking at the dew point curve in figure 15. As the diagram shows, cool air can hold much less vapor before being saturated than warm air. Therefore, in order for clouds to form, air has to cool beyond the dew point associated with the amount of vapor that it holds.

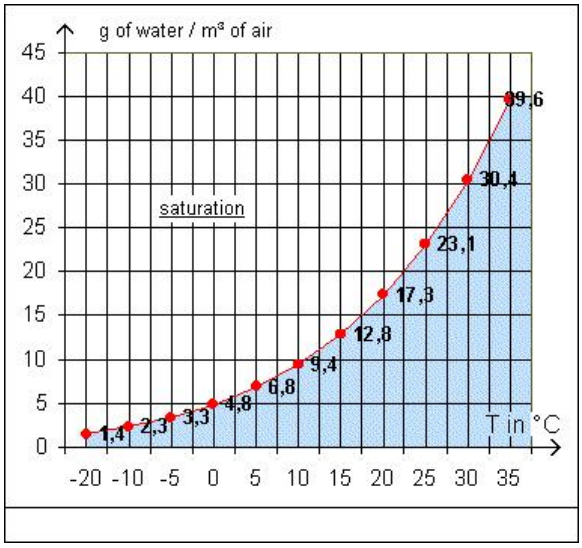

Figure 15: The dew point curve. It specifies the relation between dew point (in red) and temperature, as well as the correspondent water vapor mixing ratios

3.3 Pressure and Temperature

There are several processes that can cause a change in the temperature of air parcels. For the formation of clouds, the most important process consists in a change of altitude. As is common knowledge, air temperature decreases with increasing altitude. This is due to the decrease of air pressure.

The air pressure experienced at a certain point corresponds, roughly, to the weight of the column of air above said point, multiplied by gravity. This explains why the air pressure decreases with height. Since at higher altitudes there is less air weighing down on it, the pressure at the point is less than in lower altitudes. An important property of air is that it is a compressible gas. Its volume thus changes according to different pressure conditions. This change in volume is also responsible for the change in temperature. Between the air molecules exists a force that draws them together. When the air expands, energy is required to overcome this force.

This energy is taken from the latent energy of the air, stored as heat. The air therefore cools as it expands. When it is compressed, the process is reversed. The same principle is used, for example, in diesel engines, where the mixture of air and fuel is compressed heavily until it explodes.

3.4 Atmospheric Stability

We now know why rising air parcels tend to cool, thus causing condensation. However, in order to simulate the cloud formation process, it has to be known at what rate rising air parcels cool down. As a rule of thumb, air cools at a rate of just about 1° C per 100 meters as long as it is not saturated. In thermodynamics, this is referred to as "dry adiabatic cooling", and it can be considered somewhat of a "standard behavior" [9].

When an air parcel is saturated (e.g. its relative humidity passes 100%), some of the vapor condensates. During the process of condensation, latent energy is released in the form of heat, thus warming the air parcel. This can be explained as follows: The evaporation of water (the change from liquid to gaseous form) requires energy (usually from the sun in the form of heat). This energy remains "stored" in the vapor as latent energy. When the vapor condensates this latent energy is released again as heat. As a consequence of the release of latent heat during condensation, the cooling rate of a saturated air parcel is lower than an unsaturated one. This is refered to as "wet adiabatic cooling". The wet adiabatic cooling rate however, is not constant, since it depends on the amount of vapor that condensates and on the temperature. An average value is 0.6 °C per 100 meters.

If and how far an air parcel rises does not only depend on its own temperature but mostly on the temperature of the air that surrounds it. If the surrounding air is cooler, an air parcel will rise, if it is warmer, it will descend. Therefore the temperature distribution of the atmosphere plays an essential role in the cloud formation process.

The Earth's atmosphere is divided into several layers. Clouds can only form in the bottom layer, the troposphere (with a few rare exceptions). It is limited at the top by the tropopause, an imaginary line that also marks the beginning of the ozone layer. As the ozone layer absorbs energy in the form of ultraviolet radiation, the air temperature begins to rise again, thus forming an impassable barrier for air parcels coming from below. Therefore no clouds can form above the tropopause.

3.4.1 Stable Atmosphere

In the troposphere the temperature distribution can differ quite considerably from the adiabatic ideal. A temperature drop of less than 1° C per 100 meters is referred to as underadiabatic. This is very often the case. In fact, the International Standard Atmosphere (ISA), a standardized atmospheric

model of the Earth's atmosphere, shows only a temperature lapse of 0.65 ° C per 100 meters. This figure can be thought of as a general average. Especially in the winter it is even possible that the temperature increases with altitude, at least for a certain part of the troposphere. This phenomenon is called an inversion.

As a consequence of under adiabatic stratification, the troposphere becomes very stable, since it impedes vertical air movements. The left hand side of figure 16 demonstrates this: When an air parcel is lifted to a given altitude, it cools adiabatically. However, since the surrounding atmosphere shows an underadiabatic temperature gradient, it cools slower than the air parcel and the air at the new altitude will be warmer. As a consequence, the air parcel will not rise further and even move downwards until it reaches its original altitude where its temperature is the same as the surrounding air. Since there are therefore only very few vertical air movements, the atmosphere and the weather become very stable. As far as clouds are concerned, a stable atmosphere manifests itself in stratus or cirrus clouds (the next section explains the different cloud types in more detail) that have only a limited vertical extension [9].

3.4.2 Unstable Atmosphere

The atmosphere can also exhibit an overadiabatic temperature gradient. This means that the temperature drops faster than adiabatically with increasing altitude. The result is contrary to the aforementioned case of a stable atmosphere: instead of hindering vertical air movements and turbulence, an overadiabatic temperature gradient even increases them, thus leading to an unstable atmosphere. This is shown on the right side in figure 16. Such unstable conditions often occur in hot and humid climates. In our moderate climate zone this mostly happens in the summer: As the sun heats the ground, the air directly above it is also heated and starts to rise. As this process continues, the atmosphere becomes increasingly unstable as the temperature gradient between lower and higher altitudes grows. Since the unstable atmosphere furthers the rise of heated air parcels, clouds with a large vertical extension, like cumulus or even nimbocumulus develop that often lead to heavy rainfall and even thunderstorms.

It has to be noted that the temperature gradient of the troposphere is generally not constant. The atmosphere may be unstable at low altitudes and exhibit stable conditions further up, all at the same time. Several cloud layers of different cloud types are often a sign that the stability conditions vary at different altitudes. In general, the lower layers of the atmosphere are the most susceptible to becoming unstable because the ground, heated by the sun, can drastically change the temperature gradient.

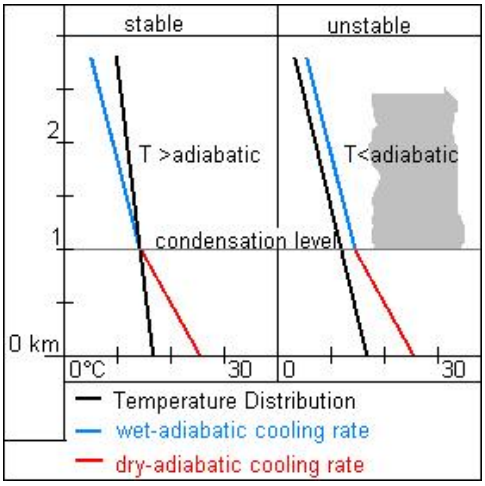

Figure 16: Stable and unstable atmospheric conditions. The black line describes the temperature gradient of the atmosphere.

3.5 Cloud Classification

When one glances at the sky and observes the myriad of different shapes and appearances of clouds, it seems at first almost impossible to consistently categorize them. From puffy summer clouds that resemble cotton candy, to the gray-in-gray that seems to cover the entire sky on a dull winter day, from dark, menacing thunderstorms to light, translucent high-altitude clouds, the dazzling array of shapes and sizes that clouds exhibit seems limitless.

The classification that is used today dates back to the beginning of the 19th century [10]. Today, it is managed and constantly updated by the World Meteorological Organization(WMO), a sub-organization of the United Nations. The International Cloud Atlas, published by the WMO, serves as an international guideline for the classification and description of clouds observed all over the world [10].

3.5.1 Classification according to WMO

In general, there are ten different types of clouds, although these types can be further classified according to their shape.

The first criterion to distinguish different categories of clouds is the altitude in which they appear. Since clouds can exist in all altitudes up to the tropopause, the distinction between different cloud families is made according to the physical state of the water they consist of. High altitude clouds are called “Cirrus” (from the Latin word for hair locks) and they consist only of ice crystals. In moderate climate zones, they can appear at altitudes between 5 and 13 km.

Medium altitude clouds have the prefix “Alto-” (lat. Altus = high). They can show up between 2-7 km above the ground and contain ice crystals as well as water droplets.

Low altitude clouds can be found from the ground up to altitudes of about 2 km. They are designated with the word “Stratus” and they consist only of water droplets.

A fourth category is that of clouds which exhibit a strong vertical extension. These clouds can span more than ten kilometers in height and are not limited to a certain altitude. One example for this cloud type are the impressive cumulonimbus, or thunderstorm clouds.

The second criterion in this classification refers to the general shape of the clouds. Cumulus clouds are heap-shaped, while stratus clouds form a more or less uniform layer. The final designation for a certain cloud type is then a combination of these criteria: stratocumulus are low, heap-shaped clouds consisting of condensed water, cirrostratus are a uniform layer of high-altitude ice crystal clouds. The different cloud types are shown in figure 17.

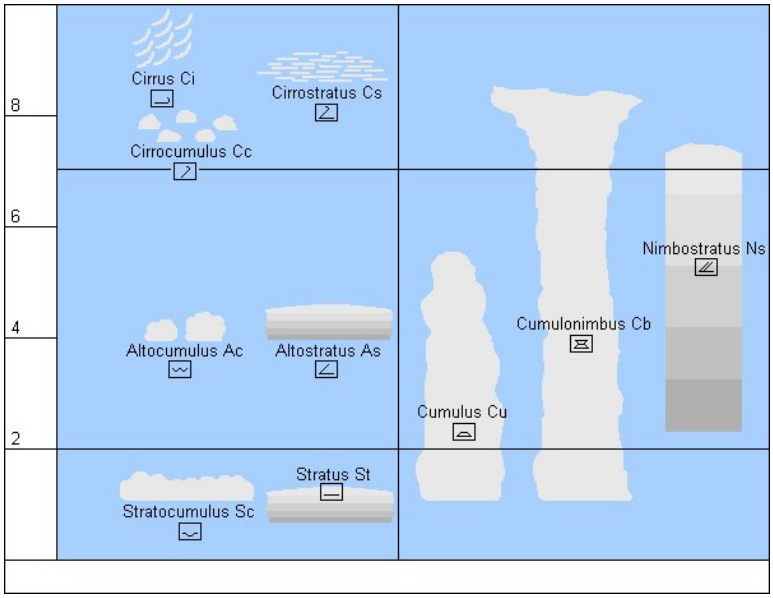

Figure 17: Cloud classification according to altitude, with official abbreviations and symbols

3.5.2 Classification according to formation process

Since we are trying to simulate the formation of clouds, it is more useful to categorize clouds according to the processes that are responsible for their existence. In general, clouds form due to vertical air movements. The

causes for these movements differ, however.

Convective clouds form as a consequence of the heating of surface air. This happens when the sun heats the ground. Parcels of heated air then start to form over the heated surface and subsequently begin to rise. Their upward velocity can be quite high, reaching up to 30m/s under very unstable conditions [9]. Once they reach higher altitudes, the vapor contained in them starts to condensate and form clouds. The condensation process releases latent heat, thus warming the parcel even further and fueling its ascension. This is the reason why cumulus clouds always grow upwards. If the atmospheric conditions are right, e.g. if the atmosphere is unstable and the ground temperature is high enough, as is often the case in the summer, cumulus clouds can keep growing upwards, eventually forming cumulonimbus clouds that can bring strong precipitations, violent gusts of wind and thunderstorms.

Another important mechanism for cloud formation is the general temperature change caused by incoming cold or warm fronts. A warm front arrives first in high altitudes because the warm air glides on top of the colder air masses. The warm front gradually pushes away the colder air, until the warm air masses reach the ground. In profile, an approaching warm front takes on the form of a wedge (see figure 18). As a consequence of the warm air being lifted, an incoming warmfront usually reveals itself by an increasing number of cirrus clouds. As time passes and the warm air gradually replaces the cold air, the cloud layer eventually becomes thicker and grows downwards, transforming into Cirrostratus, Altostratus, and finally Nimbostratus, that can bring abundant and long lasting rainfalls.

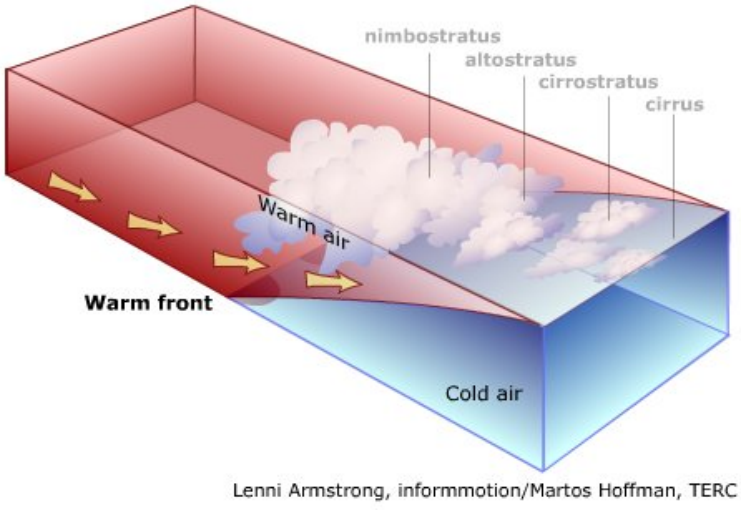

Figure 18: Schematic view of a warmfront

In contrast, an incoming cold front changes the weather quite dramat-

ically and within a short period of time. The dense cold air advances near the ground, causing the warmer air to rise rapidly. In consequence, the atmosphere destabilizes, leading to strong vertical air movements and convective clouds that eventually transform into cumulonimbus. Therefore, a cold front often causes severe thunderstorms accompanied by torrential rainfalls (see figure 19).

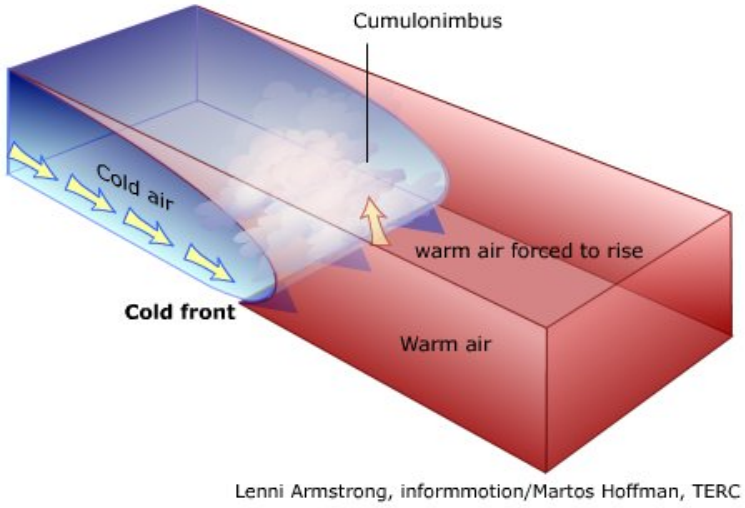

Figure 19: Schematic view of a cold front

A third important mechanism leading to cloud formation is referred to as "orographic lifting". This happens when air masses are pushed up by terrain, such as mountains. Due to the forced change in altitude, vapor condenses and clouds begin to form. This is the reason why we can often see mountains shrouded in clouds on an otherwise cloudless day. It also explains, why the weather situation in the mountains can change so quickly and seemingly without warning.

4 The Physics of Cloud Formation

As shown in the previous chapter, the formation of clouds depends on many different factors of different dimensions. High or low pressure systems can span thousands of kilometers, while microscopically small particles contained in the air and locally very limited turbulences play an equally important part in cloud dynamics. The number of contributing factors is so large and they are often so closely interrelated that certain simplifications and approximations have to be made in order to achieve a performance that allows interactivity.

The following chapter will identify and explain the most important physical laws and concepts that are used in this model for cloud simulation. In this, it closely follows the work of Harris [8] and Overby [16].

4.1 Fluid Dynamics

Clouds form due to movements of air in the atmosphere, and it is therefore the most important task of any physics-based cloud simulation to model these airflows. The physics discipline that deals with the flow of liquids and gases - fluids - is fluid dynamics. A scientifically accepted model for fluid flow are the *Navier-Stokes equations*[20], a set of nonlinear partial differential equations. More precisely, the following equations model *incompressible* fluid flow. This might at first appear to be an odd choice, since air is obviously a compressible fluid: Its density changes with pressure and temperature. In the atmosphere however, this change is very small because the air can move freely, and it is therefore a common simplification to consider air as an incompressible fluid [15]. This has also technical reasons, since the models for compressible flow only allow small time steps in order to maintain accuracy and stability [16].

Basically, the Navier-Stokes equations are nothing but the application of Newton's second law ($\vec{F} = m\vec{a}$) to fluids [1]. Compared to other types of physically-based simulations such as rigid body or particle systems however, in fluid dynamics, the physical processes are modeled from a different perspective. An intuitive approach for cloud simulation would be a particle system, where every particle represents a small amount of water or vapor. Each of these particles would then be characterized by its position and velocity, and they would be tracked through simulation space as time progresses. This is called the Lagrangian viewpoint [1]. Although being less complicated, the problem with this approach is that it would require a prohibitively high number of particles in order to achieve satisfactory results. Fluids are modeled using an Eulerian approach: The simulation space is discretized into fixed voxels and the simulated quantities are sampled at discrete grid locations. This leads to the Navier-Stokes equations. They describe the behavior of fluid at one point in a fluid volume at a given

time.

The most important quantity to represent is the velocity of the fluid, because velocity determines how the fluid moves itself and the things that are in it, like vapor, smoke, or temperature. The Navier-Stokes equations are therefore expressed in terms of velocity, $\vec{u}$. The other basic quantity needed for the simulation of incompressible fluid flow is pressure, p . It appears in the momentum equation (see equation (1)), and it is fundamental in ensuring the incompressibility of the fluid (see equation (2)).

The basic idea of fluid simulation is to start with an initial configuration (*t=0*) and then move it forward in time. At every time step, and based on the results of the previous time step, the velocity for every voxel has to be determined correctly. Once the velocity field has been calculated, the other quantities (vapor, temperature, etc...) are moved through the grid according to the state of the velocity field. As a final step, the incompressibility of the final velocity field has to be ensured.

4.1.1 Momentum Equation

The first Navier-Stokes equation is the momentum equation, and it describes how the velocity evolves at a sampled grid location over a time step t . The right hand terms therefore all represent accelerations:

$$\frac{\partial \vec{u}}{\partial t} = -(\vec{u} \cdot \nabla)\vec{u} - \frac{1}{\rho}\nabla p + \nu \nabla^2 \vec{u} + \vec{f} \tag{1}$$

In three dimensions, $\vec{u}$ consists of three components, *(u, v, w)*. ρ represents the density of the fluid. Since we assume an incompressible fluid, ρ is actually a constant. p stands for pressure, ν is the viscosity of the fluid. $\vec{f}$ summarizes all other forces that affect the velocity of the fluid (such as wind, gravity, buoyancy, etc.).

The ∇-operator is called the *nabla-* or *del*-operator. It designates the gradient (if applied to a scalar quantity), the divergence (if applied to a vector quantity), or the Laplacian (if applied to the gradient of a scalar quantity). Figure 20 shows an overview of the uses of the nabla-operator with respect to the Navier-Stokes equations. They will be reviewed again in chapter 5.

A helpful approach to the comprehension of the Navier-Stokes equations is to look at each term separately. Each one corresponds to a physical concept that adds to the total fluid flow.

Advection The first term on the right hand side of equation (1) is the so-called *advection term* . Any quantity or object submerged in the fluid is carried along by the fluid's velocity field (imagine pouring milk in your coffee). This also holds true for the velocity: it moves along itself. This explains why a fluid keeps moving even when there are no more forces being applied. This term represents this *self-advection* . The fact that the velocity $\vec{u}$

Operator	Type	Definition	Finite Difference Form
Gradient	Scalar	$\nabla p = \left(\frac{\partial p}{\partial x}, \frac{\partial p}{\partial y}\right)$	$\nabla p = \left(\frac{p_{i+1,j} - p_{i-1,j}}{2\delta x}, \frac{p_{i,j+1} - p_{i,j-1}}{2\delta y}\right)$
Divergence	Vector	$\nabla \cdot \mathbf{u} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}$	$\nabla \cdot \mathbf{u} = \frac{u_{i+1,j} - u_{i-1,j}}{2\delta x} + \frac{v_{i,j+1} - v_{i,j-1}}{2\delta y}$
Laplacian	Scalar	$\nabla^2 p = \frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2}$	$\nabla^2 p = \frac{p_{i+1,j} - 2p_{i,j} + p_{i-1,j}}{(\delta x)^2} + \frac{p_{i,j+1} - 2p_{i,j} + p_{i,j-1}}{(\delta y)^2}$

Figure 20: Use of the Nabla-operator in fluid dynamics, courtesy of Mark Harris [8]

appears twice in the advection term makes it non-linear and therefore more complicated to solve.

Pressure The next term is called the *pressure term*. As opposed to rigid bodies, an acceleration applied to a fluid does not instantly propagate through the entire fluid because its molecules are free to move. Instead, pressure builds up and the areas close to the force are accelerated first. The term "pressure" in this context can lead to confusion, since it does not refer to air pressure, a quantity that is also used in the cloud simulation. In the context of the Navier-Stokes equations, only the pressure gradient is required. It can be thought of as "whatever it takes to keep velocity divergence-free" [1]. It is also used to enforce boundary conditions.

Diffusion The third term of the momentum equation is the *diffusion term*. A fluid in motion does not keep moving indefinitely when there are no more forces being applied, and different fluids require different forces to create any movement. A measure of how much a fluid resists to flow is its viscosity, ν. In a high-viscosity fluid such as honey or ketchup, velocity diffuses much more rapidly than in low-viscosity fluids such as water or even air.

External Forces The last term accounts for all other forces that have an effect on the velocity of the fluid. A distinction has to made between global forces (such as gravity) that are applied equally to every grid cell, and local or body forces, that only affect a limited number of voxels [8].

4.1.2 Continuity Equation

The second Navier-Stokes equation is the continuity equation. As mentioned before, air is assumed to be an incompressible fluid, and the momentum equation enforces this condition.

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

When a fluid is incompressible, it means that its density is constant. This is the same as saying that mass is conserved. The continuity equation states this in terms of velocity: In order to achieve incompressibility/conserve mass, the velocity $\vec{u}$ for every grid cell has to be such that the total (negative) flow out of a cell and the total (positive) flow into a cell cancel each other out. In other words, the *divergence* of each grid cell has to be zero. In the next chapter, a method will be presented that removes any divergence and solves the pressure term of the momentum equation at the same time.

4.1.3 Dropping Viscosity

Viscosity is a measure for the thickness of a fluid. While it plays an important role in simulating highly viscous fluids like honey or when modeling very small scale flows, it only contributes very little to the velocity of an almost inviscid fluid like air. At the velocities that arise when simulating clouds, its effect is so small that it is common to neglect viscosity entirely [1], and it is therefore assumed to be zero. Although thus introducing certain inaccuracies, it has the advantage that the entire diffusion term of the Navier-Stokes momentum equation drops out, in turn speeding up the computation. This leaves the simpler *Euler equations of incompressible inviscid fluid motion*:

$$\frac{\partial \vec{u}}{\partial t} = -(\vec{u} \cdot \nabla)\vec{u} - \frac{1}{\rho}\nabla p + \vec{f} \tag{3}$$

$$\nabla \cdot \vec{u} = 0 \tag{4}$$

4.1.4 Other Quantities

The basic quantities that describe fluid flow are, as shown, its velocity and pressure. While they completely describe fluid flow, what we are actually interested in are the different quantities that are submerged in and transported by the flow. What these (scalar) quantities are depends on the application: a smoke simulation such as [5] introduces a scalar smoke density value as well as a temperature value for each grid cell. For a cloud simulation, the relevant parameters are temperature, water vapor and condensed water, with the latter being the quantity that is finally rendered to the screen. Abstracting away from what it actually represents, the application of the Navier-Stokes (or Euler) equations to any scalar quantity is straightforward. Let q be a scalar quantity submerged in the fluid flow:

$$\frac{\partial q}{\partial t} = -(\vec{u} \cdot \nabla)q + \kappa \nabla^2 q + S \tag{5}$$

The similarity to the momentum equation (eq. (1)) is quite obvious. The first term formulates that the quantity q is transported (or *advected*) along the velocity of the fluid. The second term says that q diffuses according to some constant rate κ, while S stands for any source that adds to the quantity (such as a chimney or the tip of a cigarette, in the case of a smoke simulation).

4.2 Thermodynamics

Thermodynamics is the physics discipline that concerns itself with the relation between energy and its conversion into heat and work, in dependence on pressure and temperature.

As has been shown in chapter 3, temperature is one of the most important quantities in cloud dynamics. The dewpoint, as well as the development of the buoyant forces that are responsible for vertical air movements depend directly on temperature.

4.2.1 Ideal Gases

One of the most important physical laws for the simulation of cloud dynamics is the *ideal gas law*. An ideal gas follows the ideal gas law and we treat air as such. Although in reality air consists of a mixture of several different gases. For the purposes of this thesis it will be assumed that the atmosphere is composed only of water vapor and dry air. Both these components are ideal gases[8]. The ideal gas law states the following:

$$p = \rho R_d T \tag{6}$$

p is the pressure, ρ is the density of the gas, and T is temperature. R_d is the specific gas constant for dry air, defined as $287 Jkg^{-1}K^{-1}$ [8]. This means that if either pressure, density, or temperature is constant, the other two are directly proportional. Since constant density is assumed, temperature has to decreases with pressure and vice versa.

4.2.2 Temperature Lapse Rate

The stability or instability of the atmosphere (as explained in 3.4) plays a fundamental role in the cloud formation process. It is specified by the temperature gradient or temperature lapse rate, often denoted as Γ [8]. In the cloud model developed for this thesis, the temperature lapse rate is a user-specified constant that expresses how much the absolute temperature drops per 100 m, and it is uniformly applied to the entire simulation space. This constitutes a certain simplification, since in reality the temperature gradient can vary considerably depending on altitude [9]. In our atmosphere this value varies usually between 0.55 °C per 100 m and 0.95 °C per

100 m , where the former describes an extremely stable, the latter an extremely unstable atmosphere. The standard atmosphere specifies the mean temperature lapse rate as 0.65 [9].

Based on the temperature lapse rate and the ground temperature specified by the user before starting the simulation, an initial temperature distribution for the simulation space can be computed.

4.2.3 Air Pressure

While the temperature in the atmosphere decreases more or less linearly with height (at least, this assumption is made in the simulation), air pressure reduces exponentially (see figure 21), as follows from the ideal gas law. Furthermore, since constant density is assumed, air pressure can be computed based on temperature only (and a few physical constants). The formula used here for computing the air pressure was taken from [8]:

$$p(y) = p_0 \left(1 - \frac{y\Gamma}{T_0}\right)^{\frac{g}{\Gamma R_d}} \tag{7}$$

y refers to the altitude (in meters), and p_0 and T_0 refer to the air pressure and temperature (in Kelvin) on the ground. he former is assumed as 10 kPa and the latter is a user-supplied constant. Γ is the temperature lapse rate (see 4.2.2), g is gravitational acceleration (defined as $9.81 m/s^2$) and R_d is the specific gas constant (see 4.2.1).

The air pressure is needed to compute the potential temperature (see next section) and is also required to calculate the saturation point of the air.

4.2.4 Temperature

Absolute temperature is a somewhat problematic quantity. According to the ideal gas law, the temperature depends on the pressure. It is known however, that air pressure reduces with height. Thus, if one wanted to compare temperature values at different altitudes, one would always have to consider the pressure in order to have any measure of energy. A way around this problem is to use potential temperature instead of actual temperature. The potential temperature θ of an air parcel is defined as the temperature that it would have if it were moved adiabatically down to sea level.

$$\theta = \frac{T}{\Pi} = T\left(\frac{\hat{p}}{p}\right)^{\kappa} \tag{8}$$

T designates the absolute temperature in Kelvin, p is the air pressure at the current altitude (in kPa), and $\hat{p}$ is the air pressure at sea level (which is approximated as 100 kPa). Π is called the *Exner function*.

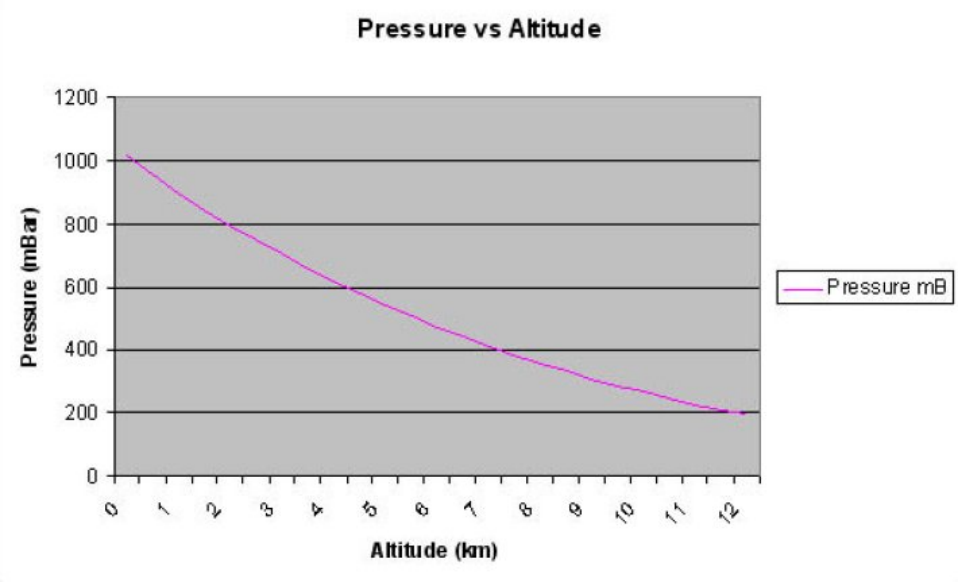

Figure 21: The air pressure decreases exponentially with altitude, figure taken from [16]

By using potential temperature instead of absolute temperature, air parcels of different altitudes can be compared regardless of the corresponding air pressure. Moreover, potential temperature is better suited to describe the behavior of rising air parcels. Air parcels rise adiabatically, which means that they cool with a rate of 1 K/100 m. However, since the calculation of potential temperature is based on adiabatic temperature changes, the *potential* temperature of rising air parcels actually stays constant under adiabatic temperature changes. Therefore potential temperature is a convenient quantity to model temperature in the simulation.

4.2.5 Buoyancy

The change in altitude of an air parcel is the result of buoyant forces. If the density of an air parcel is less than the surrounding air, an upwards (positive) buoyant force will develop, causing the air parcel to rise until it reaches an altitude where its surroundings have the same density as itself. If the parcel's density is greater, the opposite will occur because a downwards (negative) buoyant force develops.

However, this definition for buoyancy can not be applied directly to this cloud model, because it assumes a constant density. But there is a way around this problem. The main reason why an air parcel's density changes is a change in temperature. An air parcel rises if it is warmer than its surroundings, and it descends if it is colder. Therefore, the development of buoyant forces is calculated in terms of temperature, or, more precisely, vir-

tual potential temperature. By using this measure, the buoyancy increasing effect of water vapor is also accounted for. Virtual potential temperature is defined as $\theta_v = \theta(1 + 0.61q_v)$ [8].

The formula used here in order to calculate the buoyancy is based on the works of Miyazaki et al. [14] and Harris [8].

$$B = \frac{\theta_v - \theta_{v_{avg}}}{\theta_{v_{avg}}} - g \cdot q_c \tag{9}$$

θ_v is the virtual potential temperature of the current voxel, $\theta_{v_{avg}}$ refers to the average virtual potential temperature of the surrounding voxels. The second term of the equation accounts for the effect of "hydrometeors" in the air. This refers to all kinds of non-gaseous water particles contained in the air. These particles are affected by gravity, g (defined as $9.81m/s^2$), and thus exert a downward acceleration on the air parcel that counteracts the buoyancy. The only hydrometeors in this model are water droplets, expressed as the amount of condensed water q_c (in g/kg).

4.2.6 Saturation

Clouds form when the vapor in the air condensates. This depends on temperature, pressure, the amount of vapor contained in the air, and the concentration of cloud condensation nuclei. The latter are tiny particles of dust, ash, or minerals floating in the air. Since in order to condensate, water needs a surface to adhere to, the amount of condensation nuclei can play an important role in cloud formation. Without any, air can reach saturation levels of well over 100% relative humidity, while the presence of large amounts condensation nuclei can lead to condensation well below 100% saturation. However, in order to keep the model simple, cloud condensation nuclei are not modeled here, and it is assumed that condensation always sets in as soon as the relative humidity passes 100%. This leaves temperature, pressure, and vapor content as the contributing factors for condensation.

The best measure for vapor content in the air is relative humidity. The water vapor content in this cloud model is specified as a mixing ratio q_v. In order to calculate the relative humidity, the *saturation vapor mixing ratio*, q_{vs}, has to be calculated, which is the amount of vapor (per kg of air), that, at a given temperature and pressure, corresponds exactly to 100% saturation. Then the relative humidity can be calculated as $\frac{q_v}{q_{vs}}$. The formula to compute q_{vs} was taken from [8]. It depends directly on pressure and temperature:

$$q_{vs}(T) = \frac{380.16}{p} \exp\left(\frac{17.67T}{T + 243.5}\right) \tag{10}$$

T is the local temperature in °C (which is uncommon, since normally temperature values are defined in Kelvin), and p refers to the air pressure at the

current altitude.

4.2.7 Latent Heat

With the help of the saturation vapor mixing ratio, the relative humidity can be computed and it can thus be determined when condensation occurs. The amount of water that condensates is determined by the water continuity model presented in the next section. First another important phenomenon involved in cloud dynamics has to be explained: latent heat. When vapor condensates, latent energy is released in the form of heat. It is due to the release of this heat energy that the cooling rate of rising air parcels decreases once condensation sets in. The temperature in the simulation is modeled as potential temperature, θ, and therefore stays constant as long as the air is not saturated. In order to account for the release of latent heat however, θ has to be altered when condensation occurs. This is done in the following way, based on [8], where $d\Theta$ is the change in potential temperature:

$$d\theta = \frac{-L}{c_p \Pi} \cdot (-C) \tag{11}$$

L is a constant describing the latent heat released in the vaporization of water, $2.501 Jkg^{-1}$. c_p is the specific heat capacity for dry air, another constant that can be found in the physics literature [8], and its value is $1005 Jkg^{-1}K^{-1}$. C describes the condensation rate, a measure that specifies how much condensation/evaporation takes place during one time step (see next section).

4.3 Water Continuity

Water in the atmosphere constantly changes its physical state, from gaseous form to liquid, from liquid to solid, and vice versa. However, the total amount of water always stays constant. The water continuity model presented here ensures this. It also models the condensation and evaporation that result of air becoming over- or undersaturated.

The evolution of clouds is modeled by monitoring the amounts of water in its different states contained in the air, and by simulating the state changes that occur due to motion and heat. These hydrometeors can be grouped into several categories, based on particle size (if any) and physical state. Houze [11] defines up to eight different categories, from water vapor to hail. The problem is that it is difficult and computationally expensive to keep track of that many categories of water. To make things worse, they also directly depend on each other. Therefore, a simpler model is required.

The model used here to monitor and simulate the different physical states of water assumes only two categories: water vapor and cloud water, that is, water that has condensed into droplets. The latter represent

the clouds. Both quantities are represented as mixing ratios, q_v for the vapor, and q_c for the cloud water. The physical processes that link these two quantities are condensation and evaporation. They are described by a simple bulk water-continuity model presented by Houze in [11], that was also used by Harris in [8]. It consists of two simple equations that keep the total amount of water constant:

$$\frac{\partial q_v}{\partial t} = -C \tag{12}$$

$$\frac{\partial q_c}{\partial t} = C \tag{13}$$

C represents the condensation if $C > 0$, and evaporation if $C <= 0$. Both quantities, q_c and q_v, are advected by the velocity of the fluid. Thus, taking into account the advection according to equation (5) and combining both equations into one, the water continuity is expressed as follows:

$$\frac{\partial q_v}{\partial t} + (\vec{u} \cdot \nabla) q_v = -\frac{\partial q_c}{\partial t} + (\vec{u} \cdot \nabla) q_c = C \tag{14}$$

5 Implementation

5.1 Developing the Cloud Model

The cloud dynamics are modeled on a rectangular grid of dimensions dim_x+2, dim_y+2, and dim_z+2, that divides the simulation space into cube-shaped voxels of equal size. The two extra spaces in each dimension are required in order to correctly handle the boundary conditions (see 5.1.3). Translated to real world dimensions, the simulation space corresponds to a cube of 12x12x12 km. The grid is not explicitly modeled but defined by several equally sized three-dimensional arrays that contain the values for the different state variables. All state variables are defined at the center of a grid cell. The model manages and updates a total of five fields that describe the state of the simulation:

- *Velocity and pressure gradient:* These two variables result directly from the Navier-Stokes equations. The three-dimensional velocity vectors are instances of a `Vector3d` class.
- *Water vapor and cloud water mixing ratios:* The two quantities q_v and q_c represent all occurrences of water in the the model and they are a consequence of the application of the water continuity model described in section 4.3.
- *Potential temperature:* Temperature in the system is represented in terms of potential temperature, as was described in section 4.2.4.

One step of the simulation involves advection of these variables, computing buoyancy forces and updating the velocities, calculating condensation and updating vapor, cloud water, and temperature accordingly, and computing the pressure gradients and ensuring incompressibility.

Before explaining in more detail the different steps of the simulation, the initial conditions for the simulation have to be explained.

5.1.1 Initial Conditions

Since the simulation proceeds incrementally, the initial conditions have to be well posed. Also, by specifying different starting conditions, the user is given the possibility to simulate different scenarios.

The initial conditions that have to be specified before the start of the simulation concern the temperature and the humidity/water vapor distribution. The fields for the velocity, pressure gradient and cloud water are all initialized to 0. In order to compute an initial temperature distribution, the temperature lapse rate and the ground temperature have to be set. Based on these two values, the absolute temperature for each altitude (as defined by the grid spacing) is computed and stored in a look-up table of

size $dim_y + 2$. These values are then used to calculate an equally sized look-up table for the air pressure according to equation (7). Now that the absolute temperature values and air pressure are known, the field containing the potential temperature values can be initialized according to equation (8).

The initial water vapor distribution also needs to be set because an atmosphere devoid of any humidity would not be realistic. Therefore the user specifies a value for relative humidity, that is then translated into a value between 1 and 0. With the help of the values from the absolute temperature look-up table used before, the saturation vapor mixing ratio (dependent on altitude) can be computed according to equation (10). The initial water vapor mixing ratios are then obtained by multiplying the saturation vapor mixing ratio with the user-specified humidity value.

5.1.2 Solving the Navier-Stokes Equations

The Navier-Stokes equations and, derived from them, the Euler equations, have been presented in chapter 4. The question is now how to discretize and numerically solve them on the computer. Since analytical solutions typically only exist for a few simple configurations, a numerical integration technique is used to solve them incrementally [8]. The algorithm used here is based on the works of Foster and Metaxas [6] and Stam's "Stable Fluids" [20], as well as the source code published by Stam in [21].

The Euler equations representing the fluid dynamics are a sum of several terms: advection, pressure term, and external forces. As such, the easiest way to find a solution is to split them up and solve each term separately. The outcome of one step can then be used as input for the following one. One advantage of this approach is that specialized methods for each step can be applied, thus speeding up the calculations and making the simulation more stable [1].

The simulation starts in an initial configuration, t_0. Based on this configuration, the algorithm iterates through advection, external force application, has to deal with the pressure term and enforce conservation of mass (for the cloud simulation, additional steps will be needed) to produce a new configuration at time $t + \Delta t$. This then serves as a starting point for the next iteration. As a consequence, we need two fields/arrays for each of the state variables: one for the results of the previous time step, and another that holds the intermediate results of the iteration that is currently being calculated. At the end of each iteration, they are swapped, and the fields for the current state values are reset. As a result of this double storage, the memory requirements can be quite high, depending on the dimensions of the simulation space.

Solving the Advection Term The first step to solve is the advection step, where the velocity transports itself and the other quantities through the fluid. Although we are dealing with scalar quantities (heat, vapor, condensed water) as well as vector valued ones (velocity), the advection algorithm is essentially the same for both, since when advecting velocity, the x-, y-, and z-components can be dealt with separately [1].

In order to advect a quantity q through the velocity field, the advection routine requires as input a velocity field ($\vec{u}$), a time step size (Δt), and the current field quantity (q^t). The routine returns a new field $q^{t+\Delta t}$, as the quantity q is advected through the velocity field over a time step Δt.

One of the fastest and easiest approaches to solving partial differential equations is the finite difference method.

The most straightforward approach to solving the advection step is to simply write out the partial differential equation:

$$\frac{\partial q}{\partial t} = -(\vec{u} \cdot \nabla)q \tag{15}$$

Using an explicit forward Euler integration scheme for the time derivative and finite central differences for the spatial derivative (see figure 20), the new value for q at grid cell i, j, k would be calculated as follows (in one dimension):

$$q_{i,j,k}^{t+\Delta t} = q_{i,j,k}^{t} - \Delta t u \frac{q_{i+1,j,k}^{t} - q_{i-1,j,k}^{t}}{2\Delta x} \tag{16}$$

This approach follows the same idea as moving a particle around: multiplying the position of the particle at time t with the velocity times the timestepΔt yields the new position at time $t + \Delta t$.

However, this method has the significant drawback that it is unstable, meaning that the simulation will "blow up" if the magnitude of the velocity is greater than one grid cell [8][20]. This can happen easily if a large time step is chosen, if the size of the grid cells is very small, or if large velocities develop in the system.

A solution for this problem was found by Jos Stam in 1999, presented in his paper "Stable Fluids" [20], where he describes an implicit *semi-Lagrangian advection scheme* that is unconditionally stable for any time step. The basic idea is to invert the problem and use methods borrowed from particle systems to solve the advection term. It works equally well for the advection of velocity as for scalar values.

The basic idea is as follows: if the fluid were represented by particles, the advection step would be very easy to solve. Multiplying the position of the particle at time t with its acceleration times the time stepΔt would then yield the new position at time $t + \Delta t$. However, our fluid is not defined by particles but by a fixed grid, and the state values are all defined at the centers of the grid cells. Therefore we assume that there is a particle for

each grid cell that ends up exactly in the middle of it *after* a time step. This pseudo-particle can then be traced *backwards* in time (this is the reason why it is called an implicit method) by multiplying the grid position with the negated velocity times the time step. It can then safely be assumed, that the values of the state variables at the backtraced particle position will be the values at the current grid point. Of course, one such backtraced particle will almost never end up exactly at the center of a grid cell, where the state variables are defined. In that case, the values are interpolated from the neighboring cells. Figure 22 demonstrates this process graphically in two dimensions.

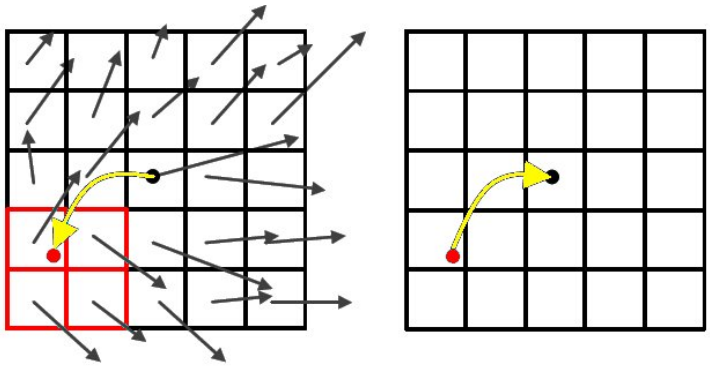

Figure 22: Tracing a pseudo-particle backwards through the velocity field and interpolating between the surrounding cells(left), copying the interpolated value to the grid position (right)

Stam [20] has shown that this advection scheme is unconditionally stable for any time step. Since it is using a Lagrangian concept (particles) on an Eulerian grid, this advection scheme is called *semi-Lagrangian*.

Adding Velocities The calculation of the force term $\vec{f}$ of the Navier-Stokes equations is straightforward. The velocity vectors that are to be added are stored in a field of the same dimensions as the velocity field. The method then simply iterates through all grid cells and adds the new values to the existing velocities, multiplied by the time step. Adding scalar quantities is done in the same way, with the corresponding scalar fields.

Enforcing Mass Conservation and solving the Pressure Term The advection step and (as we will see later) the buoyancy and water continuity computations all alter the velocity field. The result is typically a velocity field that is not free of divergence, thus incompressibility/conservation of mass, as required by the continuity equation, cannot be guaranteed. As a result, another step is required that transforms the divergent velocity field

into a divergence-free velocity field. Visually, this step is largely responsible for the swirls and vortices that fluids exhibit.

The method used here was developed by Stam [20]. It is based on a mathematical concept called the *Helmholtz-Hodge decomposition*. It states that any vector field $\vec{w}$ can be uniquely decomposed into the sum of two other vector fields: a divergence-free vector field $\vec{u}$ and the gradient of a scalar field (which is another vector field, see figure 20), ∇p:

$$\vec{w} = \vec{u} + \nabla p \tag{17}$$

This concept is actually the application of a fundamental idea from vector calculus to vector fields: A vector $\vec{v} = (x, y, z)$ can also be written as $\vec{v} = x\vec{i} + y\vec{j} + z\vec{k}$, where $\vec{i}, \vec{j}, \vec{k}$ are the unit basis vectors representing the coordinate axes [8].

By rearranging equation (17), we can see how to make the intermediate and divergent velocity field divergence-free:

$$\vec{u} = \vec{w} - \nabla p \tag{18}$$

By subtracting the pressure gradient from the divergent velocity field, we obtain a divergence-free velocity field $\vec{u}$, thus satisfying the incompressibility condition as postulated by the continuity equation. Figure 23 shows this concept graphically.

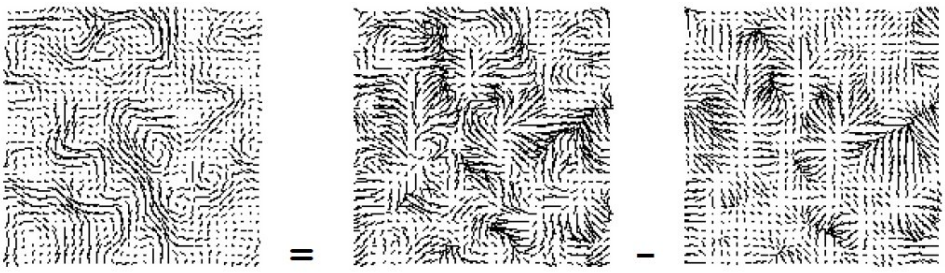

Figure 23: The divergence-free velocity field (left) consists of a divergent velocity field (middle) minus the pressure gradient field (right), courtesy of Stam [21]

Before showing how to calculate the pressure gradient ∇p, let us see how the Helmholtz-Hodge decomposition can be used to solve the pressure term of the momentum equation (equation (1)): Based on the Helmholtz-Hodge decomposition one can define an operator P that projects a divergent velocity field $\vec{w}$ onto its divergent-free component $\vec{u}$ [8]. Applying P to both sides of equation (17) yields:

$$P\vec{w} = P\vec{u} + P\nabla p \tag{19}$$

However, by definition of P, $P\vec{w} = \vec{u}$, and therefore $P\nabla p = 0$. By applying this operator to the momentum equation (equation (1)) we obtain one

compact formula that unites both Navier-Stokes equations into one single formula for the velocity:

$$\frac{\partial \vec{u}}{\partial t} = P(-(\vec{u} \cdot \nabla)\vec{u} - \frac{1}{\rho}\nabla p + \nu \nabla^2 \vec{u} + \vec{f}) \tag{20}$$

Since $P\nabla p = 0$, the pressure term drops out. Keeping in mind that the diffusion term also drops out (as shown in chapter 4.1.3) the one equation that is left describing a fluid simulation step is:

$$\frac{\partial \vec{u}}{\partial t} = P(-(\vec{u} \cdot \nabla)\vec{u} + \vec{f}) \tag{21}$$

The question is now, how to compute the pressure gradient ∇p. Applying the divergence operator to both sides of equation (17) yields

$$\nabla \cdot \vec{w} = \nabla \cdot (\vec{u} + \nabla p) = \nabla \cdot \vec{u} + \nabla^2 p \tag{22}$$

Since the continuity equation requires that $\nabla \cdot \vec{u} = 0$, this simplifies to the following Poisson equation:

$$\nabla^2 p = \nabla \vec{w} \tag{23}$$

There exist numerous methods to solve this equation. The method used in this thesis follows the one used by Stam in [21]. It is a Gauss-Seidel relaxation technique. Although there are faster methods, this method is fairly easy to implement.

Vorticity Confinement A characteristic quality of fluids are the small scale swirls and vortices, and, as mentioned before, enforcing the incompressibility of the fluid is the step that is largely responsible for creating them. However, the fluid and thus the clouds are simulated on a relatively coarse grid. Also, the advection method implemented here, although being unconditionally stable, leads to dissipation, because the interpolation step effectively takes a weighted average of values. Small scale details do not show up, because the advection step basically works as a low-pass filter (essentially this is what makes it so stable). There are several solutions to solve this problem. One possibility would be to randomly introduce rotational turbulences into the velocity field. However, this does not guarantee that the vortices are created in the right places and even viewers without a physical background notice it as unrealistic [18]. Another method, presented by Fedkiw et al. in [5], is called "Vorticity Confinement". The basic idea of this method is to find out, where these small scale turbulences and vortices have been damped out and then artificially add this lost energy back into the system [5]. Vorticity confinement was implemented in this thesis.

The first step in the calculations is to find out where vorticity occurs:

$$\vec{\omega} = \nabla \times \vec{u} \tag{24}$$

, where $\vec{\omega}$ designates a vector field describing the rotation of the velocity field $\vec{u}$, or in other words, the vorticity.

Based on the vorticity field $\vec{\omega}$, a gradient field containing the normalized vorticity location vectors $\vec{N}$ are computed, that point from lower vorticity concentrations to higher vorticity concentrations [5] :

$$\vec{N} = \frac{\eta}{|\eta|}, (\eta = \nabla|\vec{\omega}|) \tag{25}$$

The gradient field thus contains normalized vectors, that point in the direction of the strongest increase in vorticity [18]. In order to increase the vorticity, a force $\vec{f}$ perpendicular to the vorticity has to be added:

$$\vec{f} = \epsilon \delta x (\vec{N} \times \vec{\omega}) \tag{26}$$

ϵ is a scaling parameter and δx corresponds to the grid scale. This force is then added to the velocity field just like any other.

5.1.3 Thermodynamics and Water Continuity

The implementation of the buoyancy calculation (4.2.5), condensation (4.3) and latent heat (4.2.7) are basically the literal translation of the presented formulas into code. Each one of the three methods iterates through all voxels and applies the corresponding formula.

Water Continuity The water continuity is modeled by advecting q_c and q_v through the velocity field and by applying equation (14). The new respective mixing ratios are calculated as follows [8], with q'_v and q'_c describing the intermediate values after the advection step:

$$\begin{aligned} \Delta q'_v = -\Delta C = min(q_{vs} - q'_v, q'_c) \\ q_v = q'_v + \Delta q'_v \\ q_c = q'_c + \Delta q'_c \end{aligned} \tag{27}$$

This means that all the vapor q_v that condenses at a fluid cell is added to q_c and vice versa. ΔC refers to the amount of condensation during one time step. The values are stored in a three-dimensional array of the same size as the simulation space. q_{vs} is the saturation mixing ratio and computed according to formula (10). This equation requires the the absolute temperature T, which is computed with equation (8) (by solving for T and converting to °Celsius), as well as the air pressure (equation (7)).

Latent Heat Based on the results for the condensation rate C, the latent heat release can be computed and the potential temperature θ is updated accordingly. Since the potential temperature also gets advected, the full temperature update is calculated as follows, according to equation (11), and with θ' referring to the intermediate result after advection:

$$\theta = \theta' + \frac{-L}{c_p \Pi} \cdot (-C) \tag{28}$$

Buoyancy The buoyancy is calculated exactly as described by equation (9). The resulting values are stored in an array, similar to the condensation rate. Although buoyancy adds velocity, the buoyancy values are stored as scalar values and not as three-dimensional vectors. This is possible because the buoyancy only exerts a force in the vertical direction (the y-component of velocity). Since the other two components would be zero anyway, buoyancy is stored as a scalar value. They buoyancy values are later converted into appropriate vectors when they are added to the velocity field.

5.1.4 Boundary Conditions

The Navier-Stokes describe the processes that take place in the interior of a fluid. But the space in which it is simulated is limited, and therefore special care has to be taken at the limits of the simulation space.

As mentioned before, the voxel grid describing the simulation space disposes of extra grid cells in order to handle the boundary conditions, one cell in every direction. In this way, the operations that rely on values from neighboring grid cells can be carried out correctly. However, the boundary cells have to contain sensible values.

At the bottom and at the top of the simulation space, the boundaries are assumed to be solid. This might appear as an odd choice for the top boundary because in reality there exists no such solid boundary. Nevertheless, as explained in section 3.4, the tropopause and the ozone layer work in fact as a boundary as far as rising air parcels are concerned.

In terms of velocity, a solid boundary means that the velocity inside the boundary cell has to be zero. Also it has to be made sure that nothing flows into or out of it. For the velocity, this can be achieved by setting the normal component of the velocity to zero, $\vec{u} \cdot \vec{n} = 0$. In this case the normal vector to the solid boundary is simply the y-axis.

The pressure gradient also has to be zero at a solid boundary. In order to achieve this, the pressure values at a solid boundary is set equal to the closest fluid cell.

At the side boundaries, the velocity is set to zero, or, if wind is activated, to the specified wind speed. The potential temperature and water vapor mixing ratio are set to the initial, altitude dependent values.

5.1.5 Simulation Loop

The entire simulation loop consists of a total of nine steps. Each one of the steps iterates once through all voxels.

- Advection of the velocity
- Advection of θ, q_v, and q_c based on the new velocity field
- Add values from sources (wind, heat sources)
- Compute and add vorticity confinement forces
- Compute and add buoyancy
- Update q_v and q_c according to the water continuity model
- Update θ
- Set boundary values
- Ensure Incompressibility

5.2 Rendering

The outcome of the simulation loop are several fields containing the state variables. The most important one is the cloud water mixing ratio, q_c, as it represents the condensed water and thus the clouds. However, since the aim of this thesis is also to visualize other aspects involved in the cloud formation process, the other fields, especially the velocity field and the temperature field, also have to be taken into account.

5.2.1 Rendering the Clouds

The biggest difficulty in rendering clouds is accounting for the complex interactions between light and water droplets. Complicated phenomena such as light scattering and self-shadowing have to be simulated, thus dramatically increasing the computation time. The goal of this thesis is thesis however, is not to achieve photo realistic cloud images, but to visualize the physical processes involved. With this in mind, and also in an effort to increase frame rates and thus interactivity, it was decided not to include a lighting model into the simulation.

The chosen rendering method is very basic. In turn, it is also very fast, and despite its visual simplicity, it is still able to capture the dynamics of billowing clouds. The actual data that has to be rendered to represent the clouds is provided in a three-dimensional array containing the cloud water mixing ratios for every voxel of simulation space, q_c. The clouds are

rendered as simple white quadratic polygons (Quads), that are oriented towards the front. Each one is specified by four vertices, which correspond to the midpoints of the four nearest voxels. The cloud water mixing ratios defined at these points are then interpreted as opacity values, specified as the alpha component of the rgba-color. While rendering, the hardware interpolates linearly between the values provided for the corners of each quad. The basic principle is shown in figure 24.

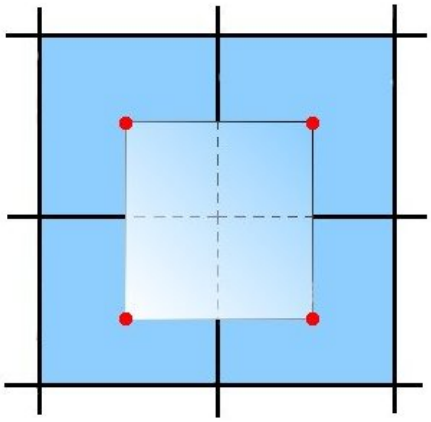

Figure 24: This image shows how the clouds are rendered (in 2D). The red dots correspond to the centers of the grid cells.

Using this rendering method, it is very important that the algorithm proceed from back to front. Otherwise, only the foremost "slice" would be rendered, covering all the others behind it.

5.2.2 Visualizing Velocity

The clouds that appear in the simulation are the result of the velocity transporting vapor, water, and temperature through simulation space. Therefore, in order for the user to gain an understanding of the processes involved, the velocity field has to be visualized in some way. The velocity field contains a 3-dimensional vector for every voxel. Each one is drawn as a straight line originating from the voxel centers. The length of each line represents the magnitude of the velocity vector. However, drawing the velocities of the entire three-dimensional field would not be useful, because there would simply be too many to deduce any meaningful information. Instead, only the velocities of one vertical slice of the simulation space are rendered at a time. The user interface lets the user decide which slice to render, and allows to change it in real time and while the simulation is run-

ning. In this way, the vertical air movements that are so fundamental in cloud formation can easily be recognized. As a further visual help, colors are used to differentiate between upwards (red) and downwards velocities (white). Upon experimenting with this technique, it turned out that the velocities that develop in the system are usually very small. Therefore a scaling factor was introduced in the rendering routine in order to better recognize high-velocity areas.

5.2.3 Temperature

Another important quantity to be visualized is the temperature. Similar to the velocity, the temperature distribution of the atmosphere is also presented as a vertical slice through simulation space. Just as with the velocity representation, the active slice can be moved back and forth along the z-axis of the simulation space.

The temperature is depicted in false-colors. Contrary to the simulation, the temperature measure used here is the absolute temperature, and the values are obtained from the potential temperature values using equation (8). Potential temperature is not visualized directly because, as explained above, it differs only slightly throughout the atmosphere.

The colors are stored in a look-up table that is created before the start of the simulation. It contains 200 color values that are obtained by respectively incrementing and decreasing the red, green, and blue components of the color. The resulting color values thus gradually change from magenta to red, orange, yellow, green, cyan, and blue, in that order. These colors are then mapped to the temperature range of the atmosphere once the simulation is initialized.

5.3 User Interface

The user interface was developed with QT, version 4.4.3. The reasons for using this toolkit are that it is free (for non-commercial applications), its platform independence, and the ease of integrating it with OpenGl. The goal when designing the interface was to make it as intuitive as possible. However, given the big amount of parameters that enter into the simulation, a basic understanding of cloud formation is still required to understand the significance of every parameter. It was therefore decided to facilitate the use of the application by letting the user choose from a series of preset scenarios. As this is only optional, advanced users can still change all the parameters manually.

The interface is divided into three areas. The biggest part is taken up by the OpenGl window displaying the simulation. Directly below it are the technical controls for the simulation, while the entire right side is reserved

for the controls that change the parameters used in the simulation. The different visualization options are also found in this area.

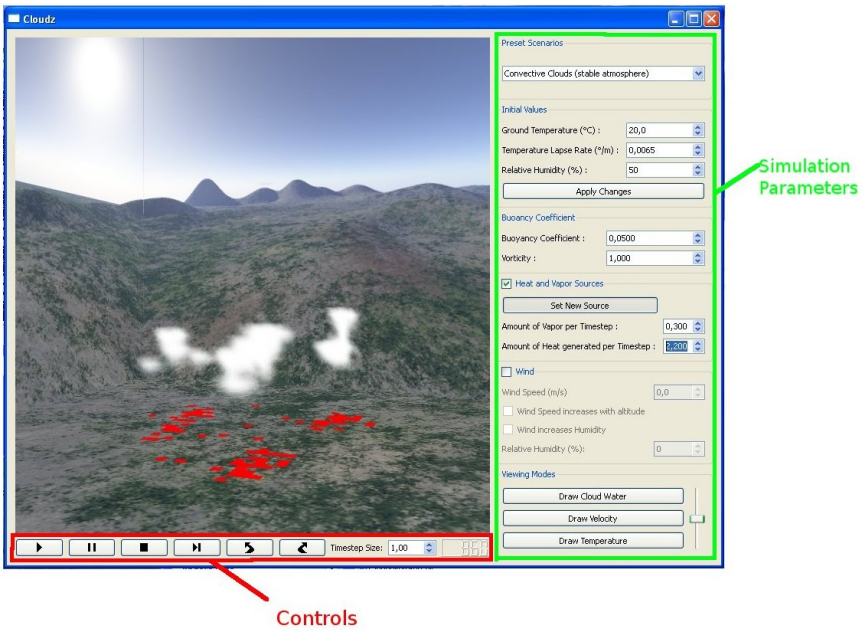

Figure 25: The user interface

Technical Controls The technical controls comprise a series of buttons that let the user start, pause, and reset the simulation, or advance it one step at a time. Further buttons are supplied that rotate the camera and set the time step size to change the speed of the simulation. A counter on the far right keeps track of the number of iterations computed.

Simulation Parameters The topmost section offers the possibility to select from a list of different scenarios. By choosing one scenario, all the other parameters are set accordingly and the simulation is reset.

Directly below are the controls to set the initial conditions for the simulation space. They include the ground temperature, temperature lapse rate, and relative humidity. Changing any of these parameters requires resetting the simulation, which is done with the button below.

The following section contains technical parameters that do not correspond to any physical quantity, but that are used as scaling factors for the simulation. They are necessary to account for inaccuracies resulting from the applied cloud model. This concerns mostly the buoyancy (as will be explained later). The other factor scales the vorticity confinement forces that are applied to the velocity field.

Below the technical parameters is the section that lets the user set heat sources at ground level. They are required for the formation of convective clouds (see 3.5.2). The idea behind this is to simulate the heated ground surfaces that lead to buoyancy and later to cumulus clouds. The user is required to set these sources by hand by clicking and dragging the mouse inside the OpenGl window and thus gains a certain control over where the clouds will appear later on. The 2D window coordinates of the mouse position are transformed into 3D world coordinates by reading the depth buffer under the mouse cursor and subsequently feeding them to an `unproject`-routine supplied by OpenGl that effectively reverses the steps of the rendering pipeline. The resulting 3D-coordinate is then used to calculate the corresponding grid cell below the mouse cursor. Selected cells are rendered as red rectangles.

The strength of the heat sources is also set in this section. The value specified here is then added at every time step to the grid cells that correspond to the user-defined heat sources.

In the next section the variables relating to wind can be set. In order to keep the interface as simple as possible, a wind direction cannot be specified and any wind therefore always blows in from the left side of the simulation space. What can be changed, however, is the wind speed, specified in meters per second. This value is then scaled based on the grid resolution and the chosen time step, and the result is added to the leftmost grid cells of the simulation space at every iteration.

Two other options allow changing the properties of the wind. The first one tries to simulate the fact that wind speeds in higher altitudes are often much higher than near the ground. If this option is chosen, the specified wind speed refers to the highest wind speed, applied to the topmost layer of the simulation space. The values for the lower altitudes are linearly interpolated down to zero at ground level.

The last checkbox assigns a humidity value to the incoming winds. This is important for the development of stratus clouds. The specified humidity value is added to the existing vapor values in the leftmost fluid cells.

The buttons on the bottom end of this section of the user interface allow to change between the different visualization options clouds, velocity, and temperature, as described above. The slider widget on the right is used to specify the z-coordinate for the temperature and velocity representations.

6 Results

This chapter will present and discuss the results of the simulation.

6.1 Convective Clouds

The user gets to choose from a selection of preset scenarios that represent different atmospheric conditions. The first two simulate the formation of convective clouds, under either stable or unstable conditions. At the beginning, heat sources must be placed on the ground to simulate heated terrain that leads to buoyancy.

The set of images in figure 26 shows what happens in the simulation under stable conditions. The temperature lapse rate was set to 0.60 and the humidity to 50%. The images show the clouds and the corresponding velocity field after 340 and 760 time steps. The grid resolution used in the creation of these images was 32x32x32:

As the sequence of images in figure 26 shows, the clouds only develop slowly. It can also be seen that clouds only develop where strong upwards velocities are present, as denoted by the long red lines in the velocity field.

The following series of images shows how cumulus clouds develop in the simulation under unstable conditions. The temperature lapse rate was set to 0.75 and the strength of the heat sources was increased. Screenshots were taken after 300 and 600 iterations.

As the images show, the results obtained with an unstable atmosphere differ quite considerably from the first set of images. As can be expected, the clouds develop much more rapidly and they grow much higher, because the unstable atmosphere encourages vertical air movements. The mushroom shape of these clouds comes into being as the rising air finally reaches an altitude of equal temperature. Since the air is not warm enough to rise further, it is pushed aside by further air masses coming from below. This behavior is somewhat consistent with the shape that thunderstorm (cumulonimbus) clouds often show, however not in this magnitude.

Upon experimenting with the application, it could be observed that the cloud simulation reacts very strongly to changes in the temperature lapse rate. While almost no clouds at all form when it is set to 0.6 and below, the clouds grow and rise almost unrealistically fast with values over 0.7. This is due to the fact that the specified lapse rate is applied throughout the entire (simulated) atmosphere, while in reality, unstable conditions usually only form up to a few kilometers above the ground.

Another factor that strongly influences the cloud growth is the strength of the heat sources that are specified by the user. The effect of setting the heat sources too strong can be observed in figure 27: Due to the strong heat sources, the buoyancy and the resulting upward velocities become very high. Upon reaching the condensation level, the upward force is even in-

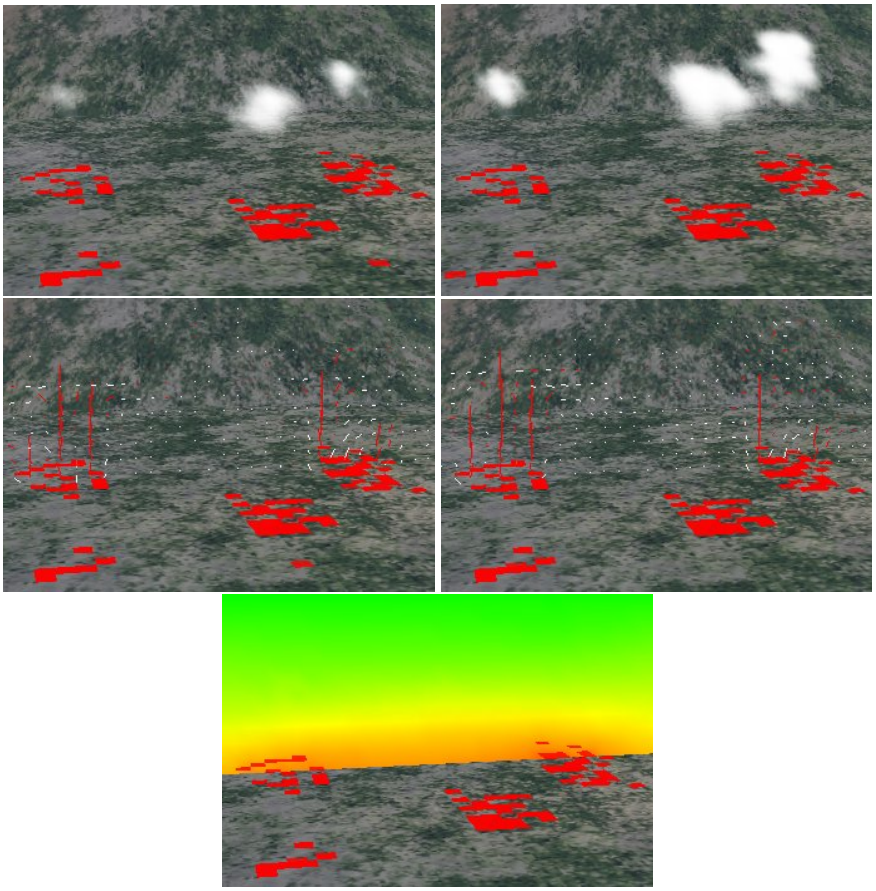

Figure 26: Cumulus clouds developing under stable conditions, and the corresponding velocity fields. Images were taken after 340 iterations (left) and 760 iterations (right). The bottom image shows the temperature distribution.

creased because latent heat is released. As a result, the air rises unrealistically fast.

However, if the strength of the heat sources is set very low, another unpleasant effect may occur. As the cloud begins to form, whole parts of it may oscillate between cloud and no cloud from one frame to another. This behavior can be explained by taking a look at the water model and the latent heat: when the vapor content in one fluid cell barely reaches saturation, a small amount of vapor condensates into cloud water, which is then rendered after the simulation loop finishes. However, the condensation process also released latent heat. But this heat is not added to the simulation until the next iteration. When it finally is added, it raises the temperature in the fluid cell enough to become undersaturated, and therefore the cloud water evaporates again, which in turn lowers the temperature.

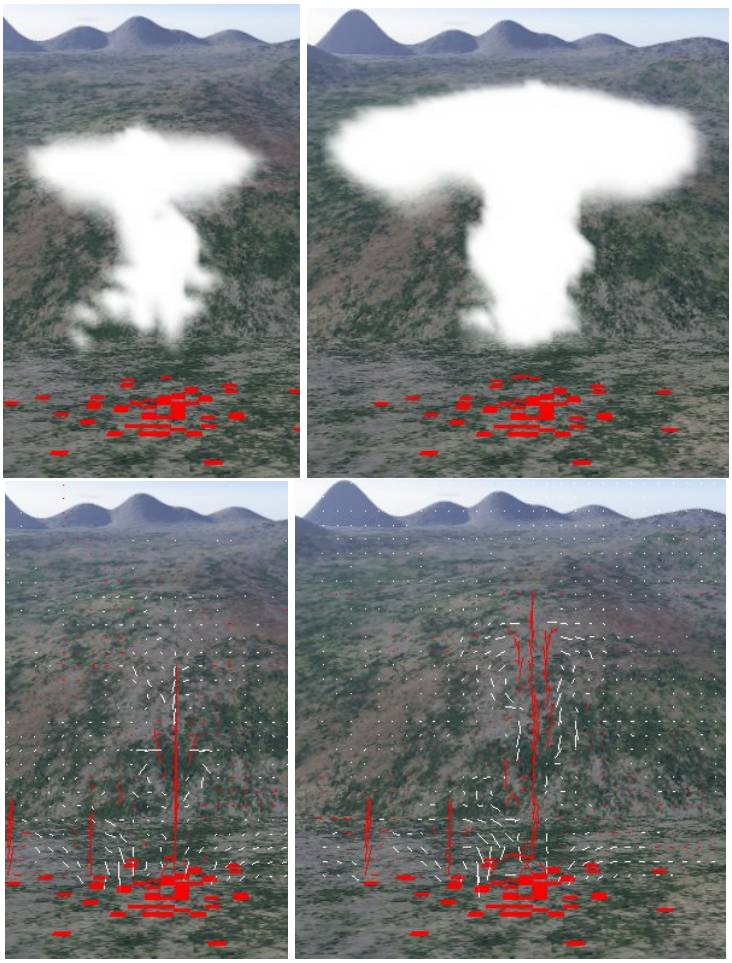

Figure 27: Cumulus clouds developing under unstable conditions, and the corresponding velocity fields. Images were taken after 300 iterations (left) and 600 iterations (right).

Because of the lower temperature, the vapor content becomes enough to start condensation again, and thus another circle begins. It was attempted to remedy this problem in the rendering routine by averaging the rendered cloud water values over two frames, but the problem still persists.

6.2 Stratus Clouds

Stratus clouds are one of the options in the preset scenarios. They can also be created "by hand" by activating the wind and setting an adequate humidity value for the wind.

The following images where created on a 40x40x40 grid. The wind velocity was set to two meters per second and the humidity of the wind to

20%.

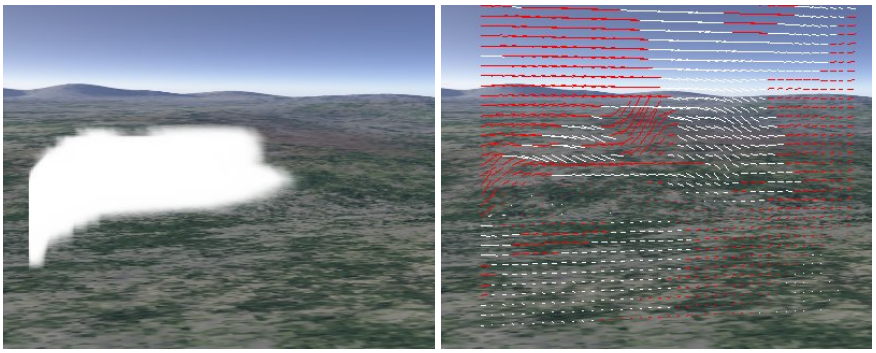

Figure 28: Stratus clouds develop as humid air is blown in from the boundary of the simulation space.

Figure 28 shows a stratus cloud and the corresponding velocity field created with this method. From an aesthetic point of view, it seems strange that the clouds just appear out of nowhere. However, there is no other way to simulate them as the simulation space is limited. The reason why the velocity field (figure 28) shows some buoyancy is that the incoming humidity condensates, thus releasing latent heat.

It has to be noted that in conjunction with wind, instabilities in the velocity field may occur. Since these instabilities begin always at the limits of the simulation space, it was concluded that they are caused by badly set boundary conditions.

6.3 Performance

Performance is always a weak spot in cloud and fluid dynamics simulation, due to the three-dimensional voxel grid. Performance drastically decreases when larger grid sizes are chosen. While a 10x10x10 grid only contains 1000 voxels, this number rises to 64000 on a 40x40x40 grid. The routines that solve the dynamics equations have to iterate several times through all voxels for each step of the simulation. On the other hand, the results of the simulation become more detailed the higher the grid resolution is set. As a consequence, a balance has to be found between performance (which usually varies depending on the machine that is used) and detail.

The simulation developed in this thesis achieves calculating one step on a 60x60x60 grid in about one second on a laptop computer containing a Pentium Mobile processor with 1.7 GHz and a Radeon 9600 graphics board. On a 30x30x30 about 8 iterations per second are calculated. This shows that the performance of the application changes linearly with the total number of grid cells.

7 Conclusion and Future Work

7.1 Summary

The goal of this thesis was to develop an interactive cloud simulation based on physical laws from fluid and cloud dynamics that can be used to visualize the physical processes involved in the cloud formation process.

Chapter 2 showed that a great number of cloud modeling methods already exist, both procedural and physically motivated. Since the simulation developed in this thesis is based on computational fluid dynamics, the most influential works in this field in relation to computer graphics were also identified.

Next, the cloud formation process in the atmosphere was explained, demonstrating the connections between air pressure, temperature and humidity. The temperature distribution of the atmosphere was identified as the most significant factor in cloud formation, before two different kinds of cloud classifications were introduced.

The following chapter was dedicated to formulating the previously described cloud formation process in terms of physics. It presented the Navier-Stokes equations of incompressible fluid motion as the method of describing the movement of air masses in the atmosphere and explained how they can be simplified. Furthermore, this chapter was used to introduce concepts from thermodynamics that are necessary for the description of cloud dynamics, and to present a water continuity model that defines the state changes of water in the atmosphere. It was also explained where and how the presented concepts abstract from reality.

In chapter 5, it was first described how the cloud model was developed based on the concepts evaluated before, identifying the temperature, water vapor, cloud water, and velocity as the state variables necessary for a cloud simulation. Next, the unconditionally stable implicit semi-Lagrangian method for the solution of the Navier-Stokes equation was presented, and it was explained how the equations for the water continuity, latent heat, and buoyant force computations are solved. All the steps were then summarized and put into context by describing the simulation loop that is executed for every step of the cloud simulation. In the following, details were given about the simple rendering method for the clouds, and the visualization methods for the velocity field and the temperature were evaluated. Finally, the user interface that controls the parameters of the simulation was explained.

Chapter six showed the results of the simulation of convective and stratus clouds and commented on the performance of the system.

7.2 Limitations of the Proposed Cloud Model

The cloud model proposed in this thesis also shows some points that are unrealistic. This is mostly due to the assumptions and simplifications made concerning the physics model, but also due to the rendering method. The following section addresses these limitations and suggests some possible methods of solutions that may be the goal of further research in the future.

7.2.1 Visual Quality

The rendering model was kept very simple, mostly as a consequence of trying to achieve high frame rates. The clouds are only rendered as white quadrilaterals, with differing degrees of translucency. Not only does this method fail to capture the interesting interactions between clouds and sunlight, but it also hides the fine details and the interesting three-dimensional structures that are supplied by the simulation. By consequently exploiting the computational power of graphics hardware, more advanced rendering methods may be possible without sacrificing too much performance.

Another drawback of the rendering method is that it does not differentiate between clouds consisting of water droplets, ice crystals, or a combination of the two. In reality they show great differences in appearance. However, even if the rendering method could differentiate between different cloud types, the water model would first have to be adjusted.

7.2.2 Physical plausibility

Most of the limitations concerning realism result from the simplified physics model. The most significant drawbacks are due to the water continuity model.

The water model applied here only knows two occurrences of water in the atmosphere, vapor and cloud water. Therefore rain cannot be simulated. As a consequence, clouds can grow much larger than in reality. When in reality a large cloud slowly disappears as it looses its humidity because of rain, in the simulation, the cloud persists and keeps growing. More advanced water models could be obtained from atmospheric sciences. However, introducing more physical states for the water in the simulation would also require to add further steps to the simulation loop as more state changes would have to be simulated. A solution could be found by, again, relying on graphics hardware to perform the cloud dynamics calculations. As the work of Harris shows, it is possible to simulate clouds on the GPU.

7.2.3 Final Conclusion

This thesis shows that it is possible to achieve a physics-based cloud simulation at interactive rates. It can also be used to visualize the most important aspects of cloud formation, and the results are physically plausible. All this, however, comes at the cost of a simplified physical model and a fairly basic rendering method.

References

[1] R. Bridson. *Fluid Simulation for Computer Graphics*. A K Peters, Ltd., 2008.

[2] Y. Dobashi, K. Kaneda, H. Yamashita, T. Okita, and T. Nishita. A simple, efficient method for realistic animation of clouds. In *Proceedings of SIGGRAPH 2000*, pages 19–28, 2000.

[3] D. S. Ebert. Interactive cloud modeling and photorealistic atmospheric rendering. In *SIGGRAPH Course. The Elements of Nature: Interactive and Realistic Techniques. Section 6*, 2004.

[4] P. Elinas and W. Stuerzlinger. Real-time rendering of 3d clouds. *Journal of Graphics Tools*, 5, 2000.

[5] R. Fedkiw, J. Stam, and H. W. Jensen. Visual simulation of smoke. In *SIGGRAPH '01: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, page pages 1522, 2001.

[6] N. Foster and D. Metaxas. Realistic animation of liquids. In *Graphical Models and Image Processing*, pages 23–30, 1995.

[7] G. Gardner. Visual simulation of clouds. In *Proceedings of SIGGRAPH '85*, 1985.

[8] M. Harris. *Real-Time Cloud Simulation and Rendering*. PhD thesis, University of North Carolina at Chapel Hill, 2003.

[9] H. Häckel. *Meteorologie*. Verlag Eugen Ulmer, 5 edition, 1985.

[10] H. Häckel. *Wetter und Klimaphänomene*. Verlag Eugen Ulmer, 2 edition, 2007.

[11] R. A. Houze. *Cloud Dynamics*. Academic Press, 1993.

[12] J. Kajiya and B. von Herzen. Ray tracing volume densities. In *Proceedings of SIGGRAPH 1984*, 1984.

[13] A. Lastra and M. Harris. Real-time cloud rendering. In *Computer Graphics Forum*. Blackwell Publishers, 2001.

[14] R. Miyazaki, Y. Dobashi, and T. Nishita. Simulation of cumuliform clouds based on computational fluid dynamics. In *Proceedings EUROGRAPHICS 2002 Short Presentations*, 2002.

[15] R. Miyazaki, S. Yoshida, Y. Dobashi, and T. Nishita. A method for modeling clouds based on atmospheric fluid dynamics. In *Proceedings Pacific Graphics*, page pages 363372, 2001.

[16] D. Overby. Interactive physically-based cloud formation. Master's thesis, Texas A&M University, 2002.

[17] K. Perlin. Hypertexture. *CGraphics*, 23, July 1989.

[18] M. Petrasch. Ein wirbelpartikel ansatz für rauch, feuer und explosionen, 2005.

[19] J. Schpok, J. Simons, D. Ebert, and C. Hansen. A real-time cloud modeling, rendering, and animation system. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2003.

[20] J. Stam. Stable fluids. In *Proceedings of SIGGRAPH 1999*, pages 121–128, 1999.

[21] J. Stam. Real-time fluid dynamics for games, 2003.

[22] N. Wang. Realistic and fast cloud rendering. *Journal of Graphics, GPU, & Game Tools*, pages 21–40, 2004.