Realistic Natural Atmospheric Phenomena and Weather Effects for Interactive Virtual Environments

Leigh McLoughlin

National Centre for Computer Animation

A thesis submitted in partial fulfilment of the requirements of Bournemouth University for the degree of

Doctor of Philosophy

Submitted: September 2012

Abstract

Clouds and the weather are important aspects of any natural outdoor scene, but existing dynamic techniques within computer graphics only offer the simplest of cloud representations. The problem that this work looks to address is how to provide a means of simulating clouds and weather features such as precipitation, that are suitable for virtual environments.

Techniques for cloud simulation are available within the area of meteorology, but numerical weather prediction systems are computationally expensive, give more numerical accuracy than we require for graphics and are restricted to the laws of physics. Within computer graphics, we often need to direct and adjust physical features or to bend reality to meet artistic goals, which is a key difference between the subjects of computer graphics and physical science. Pure physically-based simulations, however, evolve their solutions according to pre-set rules and are notoriously difficult to control. The challenge then is for the solution to be computationally lightweight and able to be directed in some measure while at the same time producing believable results.

This work presents a lightweight physically-based cloud simulation scheme that simulates the dynamic properties of cloud formation and weather effects. The system simulates water vapour, cloud water, cloud ice, rain, snow and hail. The water model incorporates control parameters and the cloud model uses an arbitrary vertical temperature profile, with a tool described to allow the user to define this. The result of this work is that clouds can now be simulated in near real-time complete with precipitation. The temperature profile and tool then provide a means of directing the resulting formation.

For Huiwen

Contents

List of Figures

List of Tables

Acknowledgements

This work represents the single largest project that I have ever embarked upon, yet I still feel as though I am only just at the beginning of my journey. Still, one chapter must end before another can begin. I would not have gotten to this stage without the kind help and support of some incredible people and I would very much like to take this opportunity to thank them.

Great thanks must be given to my supervisor, Peter Comninos, for starting me on this road, for guiding me and for never giving up on me. Without his patience and support this thesis quite simply would not exist. I would also like to thank Valery Adzhiev, my second supervisor, for his valued input and especially for his feedback on this thesis.

I am also grateful to my thesis examiners, Dan Simpson and Ian Palmer, for their comments and suggestions, which have helped shape the final version of this thesis.

I would like to express my gratitude to all my friends and colleagues at the NCCA for their advice, friendship and welcome distractions to broaden my knowledge and help me cling on to the last threads of my sanity. I would very much like to thank my fellow PhD students and good friends Eike Anderson and Steffen Engel for making the journey so interesting and full of sheep. Eike has, at times, acted as another supervisor for me and I would especially like to thank him for his guidance, kindness, uncanny paper-finding-ability and for taking the time to fully read and comment on this thesis. I am hugely grateful to Steffen for sharing his programming and system design expertise, as well as his much valued encouragement and support as a friend. I would also like to thank Olusola Aina and Eike Anderson for introducing me to LaTeX and "PhD Comics"[1]. For his artistic inputs, my sincere thanks go to Peter Hardie, especially for his advice with my early prototypes. I am also grateful to Jan Lewis for her support and encouragement over the years.

[1]http://www.phdcomics.com

I would like to thank the NCCA and Bournemouth University for funding the initial stages of this work and NVIDIA Corporation for supplying graphics hardware that greatly assisted with its completion.

Finally, I must thank my parents and family. Their support and encouragement during this long process is what has made it all possible. I would also like to thank them for their patience with my increasing obsession with clouds. I would like to thank Huiwen Zhao for being there at the end, for her warmth and encouragement, for believing in me and for giving me the final push to finish. Thank you.

Chapter 1

Introduction

1.1 Context

During our everyday lives, we see a wide variety of cloud and weather formations above us. Sometimes these displays are a collection of small innocuous clouds, peacefully drifting across a background of bright blue. At other times the entire sky can become a violent tempest, with lightning crashing above us and rain or hail whipping around us. Except for the more extreme forms, it is easy for most of us to take our weather for granted, until we are planning an outdoor event. In a virtual world, however, the absence of weather effects or their poor quality representation can be quickly noticed. Achieving visual believability in computer graphics simulations has been a driving force of industry and academia since the first computer generated images were produced. Surely, then, weather effects must play an important role in the representation of an outdoor scene.

Indeed, in films and the virtual worlds of computer games the mood of a scene can be greatly enhanced through clouds and the weather. A dark and stormy night can contain a multitude of horrors for a lone character approaching a house whilst seeking refuge from the downpour. A dull grey drizzle can be used to point out the meaninglessness of a disenchanted character's life, before the clouds part and the bright sunshine lifts his spirit. Such clichés indicate the strength and importance of weather effects in the arts.

The weather is also very important to pilots, with some weather phenomena being particularly hazardous to aircraft. Downdraught from large storm clouds can occur in what are known as micro-bursts, which are very powerful downward moving winds that can cause light aircraft to crash [Houze 1993]. Recognition of these problems and how to react to them is therefore vital to pilots to ensure their safety. Glider pilots rely on thermal updraughts to keep their unpowered aircraft in flight and knowledge of cloud formations can indicate where these thermals are. Spectacular cloud formations like the 'morning glory' in Australia can also provide unique gliding experiences, with the formation itself indicating where the updraughts and downdraughts occur [Pretor-Pinney 2006]. Realistic cloud and weather representation is therefore important to professional applications, such as flight simulators.

This project looks to develop the state-of-the-art in the subject of cloud and weather emulation for computer graphics.

1.2 Motivation and Aims

In the subject of physical science, numerical simulations are used to directly mimic reality. Once a simulation is started, its evolution relies purely on the mathematical descriptions of the physical laws it tries to emulate. The outcomes are often difficult or impossible to predict. Sometimes scientists include parameters within their equations that can be used to 'tweak' the results so that they better fit with measured data, in order to perform a more accurate simulation. The techniques become more advanced as scientists home in on reality, refining their understanding with more complex equations and calculations that take longer to execute, requiring ever more powerful hardware to run them.

Although the outputs can be interesting or exciting, within computer graphics we are not normally interested in pure physical simulations. They are generally computationally expensive, give far more numerical accuracy than we require but their outputs are often abstract and difficult to visualise directly. The key issue, however, is one of control. If the director of a film or game wants a certain feature in a specific place, how can we achieve this with a pure physically-based simulation when it might be impossible to predict the outcome? One solution, used by the ILM team which produced the visual effects for the film The Perfect Storm, was to simulate a large expanse of water with a fluid dynamics simulator. Technical directors then had to act as location scouts, searching through the virtual environment to find places suitable for their shots [Robertson 2000]. While this is a solution, it is by no means a good one. In an ideal world, the technical artist would be provided with a set of tools which would allow him or her to craft their desired result, rather than blindly running a simulation and hoping for a fortuitous output. This in itself is a complex long-term problem within computer graphics. Bringing the issue to the subject of clouds, this research must carefully consider these artistic requirements in its solutions by providing some means of influence over the results. It is primarily this topic which differentiates the goals of this work from those of the physical sciences, from which the cloud models here are derived.

Current systems within computer graphics that display atmospheric and weather effects are limited in scope. Near real-time methods are available to numerically simulate clouds [Kajiya and Herzen 1984; Overby 2002; Miyazaki et al. 2002] and others have been developed to emulate rain [Garg and Nayar 2006; Tatarchuk and Isidoro 2006] but no methods exist that link the two together. Rendering techniques exist that will visualise the outputs from meteorological simulators which include multiple water categories [Riley et al. 2003] but no techniques within computer graphics are available to generate this data. Meteorological simulations themselves are necessarily complex systems which are computationally expensive, unsuitable for an artist to set up and difficult to control. The requirement therefore exists for a lightweight, 'tailored-for-graphics' cloud simulator which bridges the gap between the basic existing models and the advanced water (including precipitation) features offered by meteorological simulations, while at the same time providing some means of influencing the resulting simulation.

The goal of this research work is to develop techniques that emulate a more complete set of dynamic cloud and associated weather phenomena which may also serve as a strong foundation for future work. The aspects of primary interest are clouds and precipitation and their linkage, i.e. to simulate clouds that are able to dynamically form and to precipitate. Further to this, with respect to the different requirements of computer graphics as compared to physical science, issues of control are to be addressed. The reason for this is to allow the techniques to be directed, to help meet artistic requirements rather than to purely obey physical laws.

1.3 Approach

The starting point for this work was an examination of previous works within computer graphics. While these varied in approach, the physically-based methods which were coupled with computational fluid dynamics arguably gave the best visual results for dynamic cloud formations. However, as previously mentioned these works did not include features such as precipitation and their actual cloud physics only covered the basics. It was also difficult to tell what the output of a simulation run would be, or where clouds would form.

These previous works are based on the real physical processes of cloud formation, so the next logical step was to examine the science of clouds and weather that is meteorology. Chapter 2 of this thesis is the result of this review effort. The aim here was to acquire a background knowledge that would allow for the development of a more advanced cloud model, but one which would be sufficiently light-weight. Surveying existing meteorological literature revealed a key work [Schultz 1995] that offered a greatly simplified water model, which is at the heart of cloud physics. This was then adopted and adapted, modified for ease of implementation and to allow some degree of control. Further examination of standard meteorological texts revealed a way of greatly influencing the vertical extent of cloud formations. A commonly used meteorological chart then inspired the development of a tool that could be used to direct this.

1.4 Contributions

The original contributions of this work are:

- The addition of multiple water types into a physically-based cloud model for computer graphics, to simulate both warm and cold (frozen) water.
- The inclusion of precipitation types into this model, to allow a simulated cloud to rain, snow or hail.
- Controls within the water model to allow some degree of artistic direction, including the ability to specify regions where precipitation may or may not fall.
- The use of an arbitrary vertical *temperature profile*, which influences the vertical formation of clouds.
- A tool, based on a meteorological chart, which can be used to set the vertical temperature profile and the initial simulation conditions. This tool allows the desired cloud base and cloud top altitudes to be directly specified. The tool also allows advanced users to create layering effects.

1.5 Thesis Overview

In Chapter 2, we will review the subject of cloud dynamics, which explains the physical process of cloud formation and evolution. We will start by identifying the various different forms that clouds can take, introducing a commonly used taxonomy that is based purely on visual appearance. This will allow us to better understand the challenge of cloud simulation by identifying our desired outputs. We will then build a mathematical model for cloud dynamics. To do this we will rely on *parcel theory,* which is a commonly used meteorological tool that aids understanding [Petterssen 1940]. It describes how small volumes, or *parcels*, of air move through the atmosphere. We will look at how to mathematically describe this ‘background’ atmosphere and the parcels that move through it. With our understanding of parcels, we will extend our model to include water. We will see how to represent water in our parcels, the states and categories it can be present in and how it can transfer between those states. Thus armed with a more complete parcel model, we will see how parcels move through the atmosphere in relation to atmospheric stability, empirical cloud formation models and finally computational fluid dynamics.

In Chapter 3, we will review previous work on cloud and weather simulation. We will briefly consider meteorological simulations, but we will be mostly concerned with computer graphics. We will generalise these into physically-based and procedurally-based simulation techniques but also consider hybrid methods. Further, we will briefly look at rendering methods, specific weather effects and more general fluid dynamics approaches.

In Chapter 4, the main work of this project will be presented. The techniques follow on from Chapter 2, where we started to build up our simulation model from meteorological sources. The model will be described in full along with the associated techniques and tools that give us a means of directing the simulation. Details of the prototype implementation will be discussed and results presented. The results will be evaluated and we shall consider the advancements of the methods and how effective the techniques are.

Chapter 5 concludes this thesis, with a discussion on the strengths and limitations of this work and an identification of various possible future directions for further exploration.

Chapter 2

Cloud Dynamics

2.1 Introduction

In this chapter we will look at the physical properties of clouds and the atmosphere in general and examine theoretical models that attempt to describe and emulate the observed characteristics. We shall begin by looking at clouds and their classification schemes, based purely on a cloud's appearance. Then we will look at the static properties of the atmosphere as a whole and consider models for describing it. We will then move onto the start of real *cloud dynamics* by examining *parcel theory*, which will allow us to describe the atmosphere in terms of the motion of small volumes of air called *parcels*. Once we have a basic understanding of parcel theory, we will look at water and see how it works with and affects parcels and the atmosphere in general. We will then look at larger-scale motions in a set of empirical cloud models, which will tie in our understanding of parcel theory to the observed characteristics of clouds. Finally we will consider fluid dynamics as a means of simulating these atmospheric motions.

2.1.1 Terminology and Techniques

This chapter relies on a number of fundamental physical laws, properties and relationships, some of which are worth briefly reviewing before they are used.

Gasses are naturally of great importance within meteorology, and there are three properties of gases that are of interest: pressure, temperature and density.

Pressure, p, is the term used to express the force, F, exerted at a normal angle by a measure of gas over its boundary of area A, in accordance with the equation $p = \frac{F}{A}$. Its unit within the SI system is the Pascal (Pa), although meteorological texts often use the millibar (mbar or mb), which has equal scale, where 1mbar = 10kPa. Many other units of measure exist, but this work will adhere to the more recent conventions as presented in the meteorological sources used.

Temperature, a property of objects that we perceive as a range from hot to cold, is a form of internal energy of a body — specifically, it is proportional to the average kinetic (movement) energy of the molecules that make up the body [Bohren and Albrecht 1998, p. 37]. It is commonly measured in degrees Celsius (°C), or in degrees Kelvin (K), which is essentially the Celsius scale shifted to start at the thermodynamic absolute zero, so that 0 K = -273.15°C. For a more detailed discussion the reader is directed to thermodynamic texts, such as Bohren and Albrecht [1998, p. 45].

Density, ρ, is mass m per unit volume V, $\rho = \frac{m}{V}$, and in SI units is measured in $kg.m^{-3}$.

The three properties pressure, temperature and density are linked together in the equation of state for an *ideal gas*, also known as the *ideal gas law:*

$$p = \rho RT \tag{2.1}$$

Where R is the gas constant per unit mass, also termed the individual gas constant [Rogers and Yau 1989, p. 2], and for dry air is approximately equal to $287\,J\,kg^{-1}\,K^{-1}$.

It is important to note that this equation is an approximation that only holds true for a theoretical *ideal* gas, which assumes that inter-molecular forces are negligible, and that the particles themselves occupy a negligible volume. For a more detailed discussion, and derivation of the ideal gas law, the reader is again directed to standard thermodynamic or physical texts, such as [Bohren and Albrecht 1998, p. 36].

Although they are more commonly used within computer graphics, it is also worth recalling some basic principles of classical mechanics, namely Newton's three laws of motion:

1. A body will not change its momentum unless a force acts upon it.

2. "If a force acts upon a body, then the resulting change of momentum is proportional to the acting force" [Benenson et al. 2006, p. 41]. This results in the following relationship between Force, Mass and Acceleration: $F = ma$.

3. For every action there is an equal and opposite reaction.

2.2 Cloud Classification

Clouds are observed to form a wondrous variety of shapes, ranging from small and innocuous cauliflower-topped shapes to large, drab homogeneous masses that drizzle incessantly. In order to facilitate the study of any such widely varying phenomena, a means of classification must be agreed upon.

A number of naming conventions for the classification of cloud forms have been proposed over the centuries, but it was the system introduced by an English pharmacist, Luke Howard, during the winter of 1802-1803 that has now become the basis of cloud classification terminology[1]. In his description, Howard introduced the use of three Latin roots, to describe three basic cloud types: cumulus (meaning heaped), stratus (meaning to spread out or cover in a layer), and cirrus (meaning a lock of hair) [Houze 1993, p. 6]. To describe clouds capable of precipitation, he added the fourth term: nimbus (meaning rain). The system then combined these word roots to build up a description of further cloud types.

The method used by the World Meteorological Organisation's International Cloud Atlas [WMO 1975], is essentially based on Howard's nomenclature. Here, clouds are first divided into three overlapping categories, or *étages*, based on their height, followed by ten unique *genera*, each containing different *species*, each of which in turn can have a number of *varieties*. Following the lead of Houze[2], we shall extend the WMO's scheme and also include fog as an additional cloud genus. Further, we shall include a fourth 'vertical' étage for the two cloud types that are capable of extending their height into all three bands. (see Table 2.1 on the following page).

It is important, however, to remember that cloud classification is an artificially imposed designation, intended purely as an aid to identification and cloud study. As such, cloud forms may change between classifications and, at times, an individual cloud may be difficult to fit neatly into a single category. Similarly, although there are links which we shall see, a cloud's classification does not necessarily reflect its method of formation. As Houze [1993, p. 3] describes, "this nomenclature makes no attempt to explain the observed clouds; it is based only on the sizes, heights, and physical appearance of the cloud".

The aim of this project is to simulate cloud forms, so it is worth considering our desired outputs. Let us now look at the four main categories and briefly note the primary characteristics of each genus. For a more detailed, but light-hearted examination of the cloud classification scheme, the Cloudspotter's Guide [Pretor-Pinney 2006] is highly recommended.

[1]As was the fashion at the time, Howard initially presented his theories at a scientific society lecture, which were later published in the *Philosophical Magazine*. Howard's life and achievements are recounted by Hamblyn [2002] and the interested reader is directed to this text, which also includes descriptions of earlier classification schemes in Chapter 6.

[2]As Houze explains, the WMO method is designed for weather observers, and classifies fog as a "restriction to visibility" [Houze 1993, p. 6]

Category (étage)	Genus	Species	Varieties
Low Level: height 0 - 2km	Fog		
	Cumulus	Humilis Mediocris Congestus Fractus	Radiatus
	Stratus	Nebulosus Fractus	Opacus Translucidus Undulatus
	Stratocumulus		
Mid Level: height 2 - 7km	Altostratus		Translucidus Opacus Duplicatus Undulatus Radiatus
	Altocumulus	Stratiformis Lenticularis Castellanus Floccus	Translucidus Perlucidus Opacus Duplicatus Undulatus Radiatus Lacunosus
High Level: height 5 - 13km	Cirrus	Fibratus Uncinus Spissatus Castellanus Floccus	Intortus Radiatus Vertebratus Duplicatus
	Cirrocumulus	Stratiformis Lenticularis Castellanus Floccus	Undulatus Lacunosus
	Cirrostratus	Fibratus Nebulosus	Duplicatus Undulatus
Vertical: height 2 - 13km	Cumulonimbus	Calvus Capillatus	
	Nimbostratus		

Table 2.1: The cloud classification scheme that shall be used in this work. This table is a reduced and slightly modified version of that from Pretor-Pinney [2006, p. 17], with the addition of Houze's temperate height data [1993, p. 7], and following Houze's lead by treating fog as low cloud, and with the addition of a vertical étage.

2.2.1 Low-Level Clouds

This is the category assigned to clouds with a base height not exceeding two kilometres [Houze 1993, p. 7]. The warm temperatures at low-altitude means that these clouds are usually composed of liquid water.

2.2.1.1 Fog

Technically, fog is a term that can be used to describe *any* cloud type that contacts the ground. However, a number of cloud types exist that are formed directly at ground level due to the ground itself, and can thus be considered as unique fog types. These include *radiation fog*, caused by radiative cooling of the ground surface; *advection fog*, where “warm air moves over a pre-existing cold surface” [Houze 1993, p. 7]; and *steam fog*, where steam rises into cold air from a warm body of water.

2.2.1.2 Stratus

These clouds are generally uniform masses, with little in the way of sharply defined features. They form flat sheets, often extremely extensive horizontally, and can be thick enough to block out the sun. They do not usually bring precipitation, although if they occur as fog the result can be drizzle or fine mist.

2.2.1.3 Cumulus

Featuring distinct edges, a flat base, and a characteristic ‘cauliflower’ top, these are perhaps the stereotypical cloud type. They start quite small, extending less than a kilometre in any dimension, but can develop into powerful *cumulonimbus*. They can occur individually, in spaced-out groups, or in closely-packed clusters. Small ‘fair-weather’ cumulus *(cumulus humilis)* have lifespans of approximately 5–40 minutes, and young cumulus are more sharply defined than older ones, which appear more ragged due to cloud erosion. [Wilhelmson and Ramamurthy ca.2008] (See Figure 2.1.)

2.2.1.4 Stratocumulus

This type exhibits the general extensive flat sheet characteristics of stratus, but consists of ‘rounded lumps’ that are similar in form to cumulus. (See Figure 2.2.)

2.2.2 Mid-Level Clouds

These clouds have a base height of 2-7km [Houze 1993, p. 7], and are generally just the mid-altitude variations of their low-level cousins. These mostly consist of

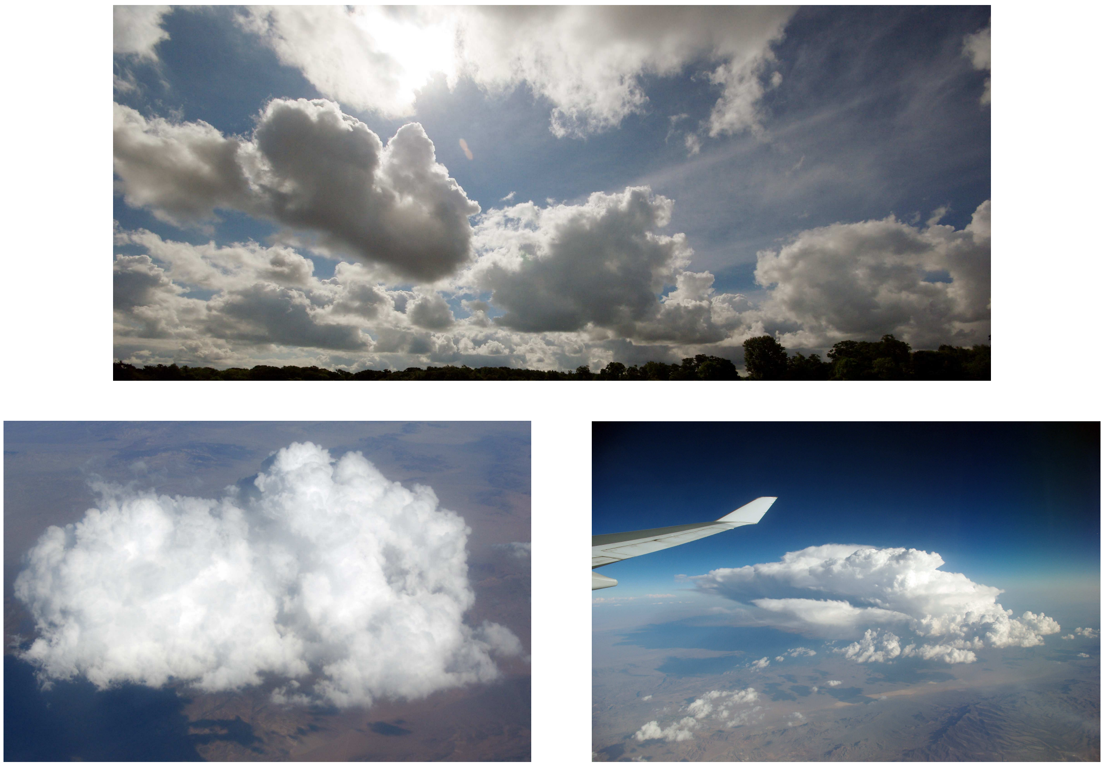


Figure 2.1: Top: low level cumulus clouds. Bottom: cumulus clouds photographed from a height of approximately 10km. The larger cloud on the right was likely to be or soon become a cumulonimbus. Photographs by the author, top June 2009, bottom pair January 2008.

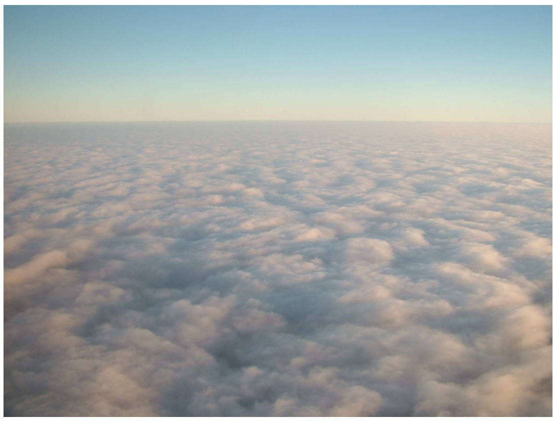


Figure 2.2: Stratocumulus as seen from above. This type is characterised by a flat layer with cumulus-like bumps. Photograph by the author, August 2007.

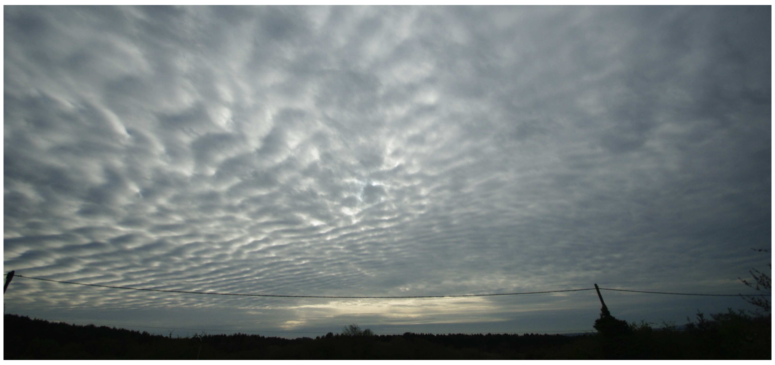


Figure 2.3: Altocumulus stratiformis, a 'lumpy layer' of cloud. Photograph by the author, April 2009.

liquid water, but can be composed of ice, or a combination of the two, depending on the atmospheric temperature profile.

2.2.2.1 Altocumulus

This type exhibits a wide range of variations, ranging from a mid-level cumulus equivalent (such as *altocumulus castellanus*) to a stratocumulus equivalent (*altocumulus stratiformis*). To aid identification, a useful 'rule-of-thumb' is that, at arms length, cumulus and stratocumulus elements are the size of the fist or larger, while altocumulus elements are the size of one to three finger-widths. [Pretor-Pinney 2006]. (See Figure 2.3.)

2.2.2.2 Altostratus

This type is similar to stratus, but is not usually thick enough to completely block the sun. Instead, they can exhibit coloured rings around either the sun or moon, called a *corona*, in a process caused by the diffraction of light by small water droplets. [Houze 1993, p. 15]

2.2.3 High-Level Clouds

These occur between a height of approximately 5km and the tropopause (the top of the layer in the atmosphere where clouds form, an altitude of about 10km) [Houze 1993, p. 7]. The temperature at these altitudes is sufficiently low that almost all cloud water is present in the form of ice crystals. Further, due to the often large particle size coupled with the low saturation vapour pressure of ice, the particles evaporate slowly and clouds are long-lived (this will be explained in more detail in Section 2.6.3). As strong high-altitude winds move the particles,

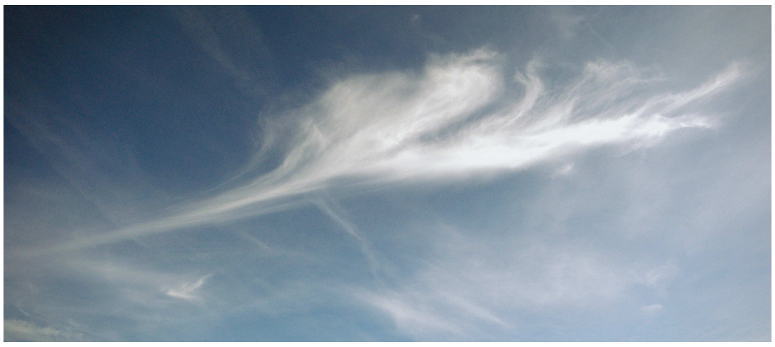


Figure 2.4: A high level cirrus element. Photograph by the author, February 2009.

the clouds form characteristic filament or wispy shapes [Houze 1993, p. 16]. High-level clouds are often caused as a result of vertical cloud formations, which bring water up from lower levels.

2.2.3.1 Cirrus

These exhibit distinctive long hair-like filaments, which are often the result of *virga* – precipitation that evaporates before it reaches the ground. (See Figure 2.4.)

2.2.3.2 Cirrostratus

Similar to their lower-level counterparts, these have a uniform sheet-like appearance. Although this type of cloud can be over a kilometre in thickness they are still relatively transparent, and their ice-crystal composition can result in a range of interesting and unusual optical effects. These include: various *halos*, distinct bright rings around the sun at specific angles resulting from certain ice crystal compositions; sun-dogs, which are a pair of bright spots that appear either side of the sun; as well as a number of rare circular and arc phenomena[3]. (See Figure 2.5.)

[3]Cloud optics is a fascinating and diverse subject, but unfortunately beyond the scope of this thesis. An excellent source of information on optical phenomena is the web resource maintained by Cowley [Cowley ca.2008], which includes many photographic references and explanations as well as HaloSim, a free ray-tracing program for simulating a wide variety of effects.

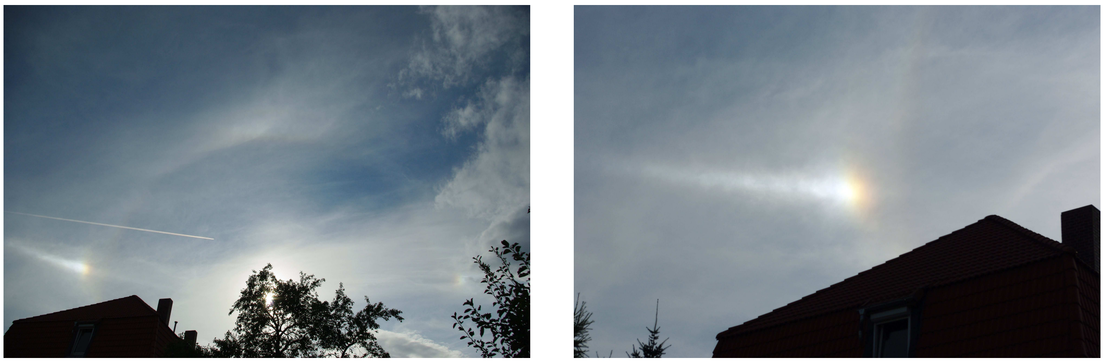


Figure 2.5: Left, sundogs and halo in a layer of cirrostratus; right, close-up of a sundog. Photographs by the author, August 2008.

2.2.3.3 Cirrocumulus

These are similar in form and variety to altocumulus, though they can sometimes exhibit the wispy hair-like filaments that are characteristic to the high-level clouds. (See Figure 2.6.)

2.2.4 Vertical Clouds

The WMO categorises clouds according to their base height, although two cloud types exist that extend across all three height zones. The 'traditional' scheme would place these two types into the low-level classification, which can be considered misleading. To aid clarity, we will separate these into their own category of 'vertical clouds'. As a result of their large vertical extent, the tops of these clouds may consist of ice while their bases remain as liquid water.

The two types within this category are the *nimbostratus* and the *cumulonimbus*. As their word-stems indicate, they are both highly precipitating cloud types and cumulonimbus is especially associated with various forms of extreme weather.

2.2.4.1 Nimbostratus

This is perhaps the extreme version of a stratus, with a large vertical extent and a characteristic uniform grey appearance. Although they rain continually, they are not associated with particularly extreme weather, and Houze [1993, p. 13] describes them as "simply a dark, rainy cloud covering the entire sky".

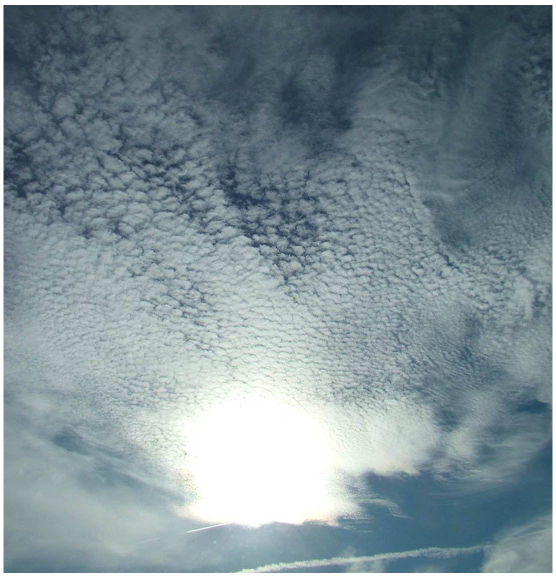


Figure 2.6: Cirrocumulus formation. Photograph by the author, August 2006.

2.2.4.2 Cumulonimbus

Cumulonimbus are of most relevance to this project, for they display all the formation and precipitation characteristics that we are aiming to emulate. These are the stereotypical thunderstorm clouds and are associated with extreme weather conditions from torrential rains to high-speed winds and devastating tornadoes, with thunder and lightning thrown in for good measure. The dynamics of thunderstorms are a complex subject and the topic of much active research, mostly in the hope of improving prediction methods that might save lives. Cumulonimbus dynamics are examined in more detail in Section 2.6.1.1 on page 50. (See Figure 2.7.)

Cumulonimbus also exhibit a number of accessory cloud varieties, which are worth mentioning. These include *shelf clouds*, which extend at low level ahead of the main cloud (see Figure 2.8a); *wall clouds*, from which tornadoes extend; and *mammatus*, so called for their breast-like protuberances on the underside of clouds (see Figure 2.8b and c). Further more, the top of a cumulonimbus may form an anvil shape, usually when it extends into an area of high atmospheric stability and spreads horizontally.

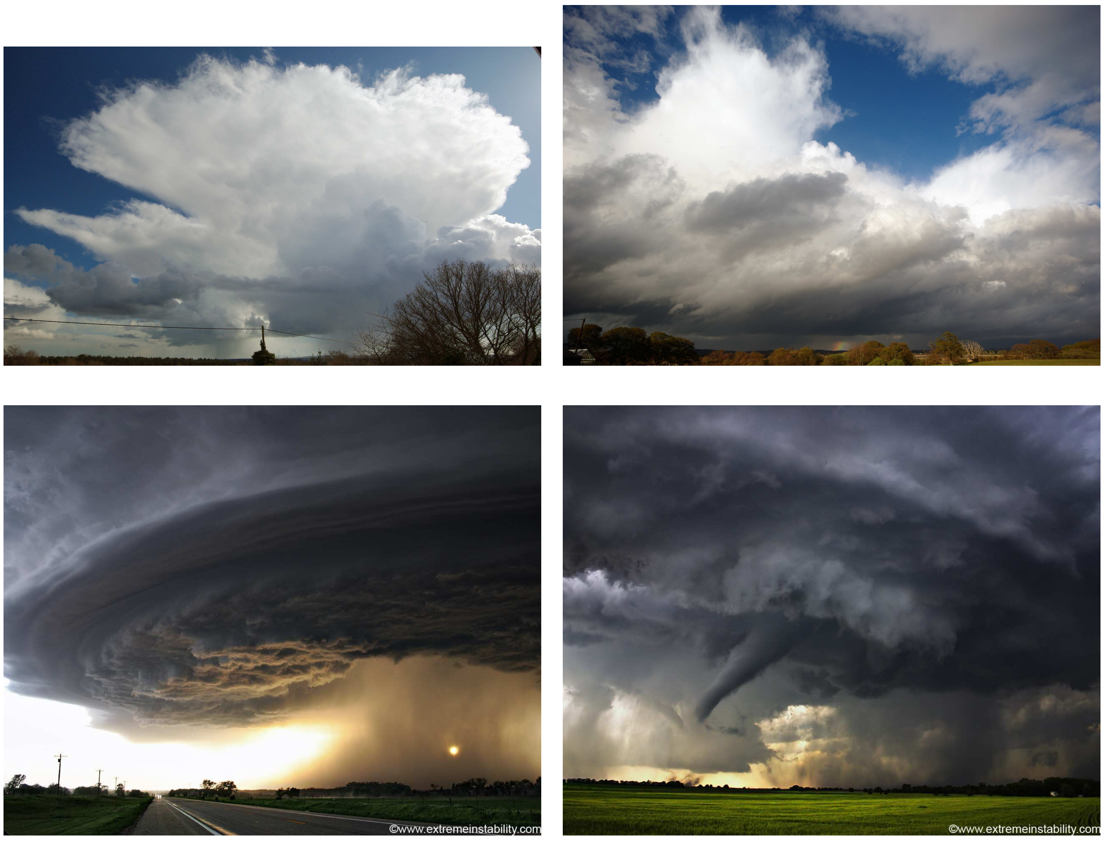
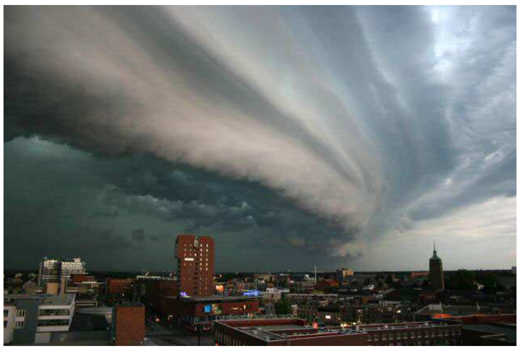
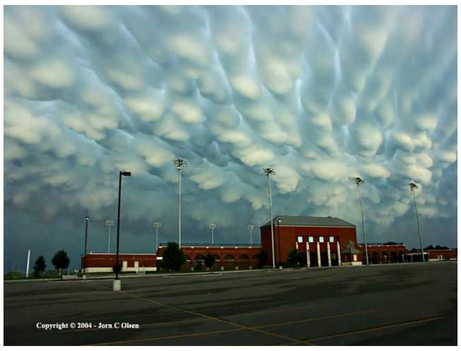


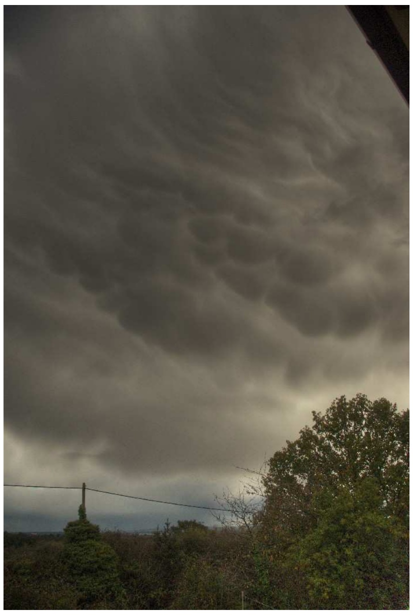
Figure 2.7: Top: typical views of cumulonimbus in the UK. Bottom: extreme cumulonimbus supercells rotate and can form tornadoes. Top photographs by the author, March 2009 and April 2012. Bottom photographs credited to Mike Hollingshead [Hollingshead ca.2009]



(a)



(b)



(c)

Figure 2.8: (a) Shelf Cloud. Image credited to John Kersthold [2004]. (b) Unusually well-defined mammatus formation. Image credited to Jorn Olsen [2004] (c) Typical mammatus formation. Photograph by the author, October 2008.

2.3 The Atmosphere and Parcel Theory

In this section we shall examine the fundamental set of atmospheric properties and its basic characteristics. We will then start to consider the atmosphere in motion by looking at an abstraction known as *Parcel Theory*, in which the movements of individual volumes of air are described as part of the atmosphere as a whole.

2.3.1 The Atmosphere

The atmosphere is a complex dynamic structure with air circulations ranging between very large (planetary) and very small (sub-metre) scales.

According to the ideal gas law, the atmosphere must consist of three well-defined and inter-related properties: pressure, temperature and density. A useful description of the atmosphere can be built up by examining these three properties.

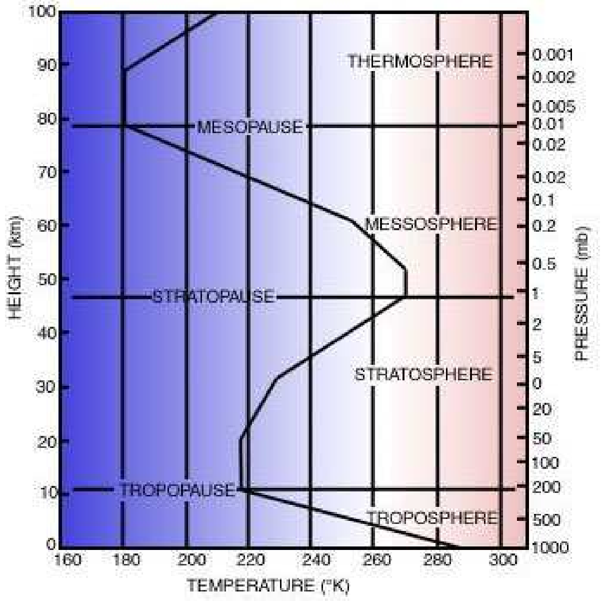


Figure 2.9: This is the “vertical temperature profile of the ICAO Standard Atmosphere”, source: [Met Office ca.2008]

2.3.1.1 Atmospheric Temperature

At ground level we are used to experiencing atmospheric temperature variations, which rise and fall as various air masses come and go. What we may not appreciate is that the temperature above us also changes. The graph of atmospheric temperature against altitude is termed a *temperature profile* or *temperature gradient*. Perhaps surprisingly, the average temperature profile shows a number of very distinct regions, which atmospheric scientists use to categorise the atmosphere into a number of discrete layers, as shown in Figure 2.9.

The region that interests us in this work is the lowest-layer, the *troposphere*, which is the first approximately 10km of atmosphere and in which the vast majority of clouds form (a few very rare types form in higher layers). Within the troposphere, the vertical temperature profile can vary, which has a great effect on weather systems. The layer above the troposphere is the *tropopause*, which has a more-or-less constant temperature throughout its height. This is a key feature which, as we will see later in Section 2.5 on page 47, inhibits buoyancy.

In its most approximated form, we can categorise the temperature gradient with the concept of a constant temperature *lapse rate*, which is simply the rate at which temperature *decreases* with height. A number of organisations and bodies have extended the idea of such approximations and designed *standard atmospheres* for various purposes, such as the testing of aircraft or the calibration of equipment. These standardisations use a set of lapse rates, or more complex equations, to define the temperature, pressure and density profiles. The ICAO Standard Atmosphere[4] gives the lapse rate in the troposphere as $6.5\,K\,km^{-1}$.

These models essentially describe the ideal average temperature gradient and, while this may be a useful starting point, it does not accurately represent a real temperature gradient at a specific time instance. For example, during the day the sun heats up the ground, which in turn heats up the air above it resulting in a warmer region at low altitude. Thus, real gradients exhibit multiple regions of different lapse-rates and are not necessarily smoothly varying. These irregularities are important to cloud formation because they affect buoyancy, the full effects of which we shall examine in Section 2.5 on page 47. It is important then to ensure that the atmospheric temperature gradient is described using a method that allows such irregularities — for example, using multiple lapse-rates or using a set of temperature values at sample points of given heights and employing some form of interpolation when querying between samples.

[4]The International Civil Aviation Organisation (ICAO) Standard Atmosphere, which defines the atmosphere up to 80km, is Doc. 7488 [ICAO ca.2008] and a table of results up to 20km may also be found in the back of Rogers and Yau [1989, p. 277]. A number of Standard Atmospheres exist from other organisations, including: The International Organisation for Standardisation's (ISO) Standard Atmosphere, which is ISO 2533:1975 [ISO ca.2008]; The U.S. Standard Atmosphere 1976, is available from NASA [NASA ca.2008]. A number of web-applets are also available on-line which allow the user to query a standard for atmospheric properties based on a given height — for example, an ICAO calculator [Frei ca.2008], and a U.S. Standard Atmosphere calculator [Digital Dutch ca.2008].

2.3.1.2 Atmospheric Pressure

Although the pressure at the Earth's surface can vary, it generally decreases with height in a regular manner as the atmosphere fades into space. This atmospheric pressure gradient can be modelled by considering the Earth to be flat and the atmosphere to be static in terms of its vertical motions. The pressure at a given point is then assumed to be caused by the weight of the air pressing down from above[5]. When the atmosphere is in such a state, it is considered to be in *hydrostatic equilibrium* [Rogers and Yau 1989, p. 28].

If we consider a small volume of such an atmosphere then, by our definition, it will have no vertical motion. According to Newton's first law of motion this means that there is no resultant force acting upon the volume, so any forces which do act upon it must be in equilibrium. There are three such forces:

- A downward force from the atmosphere above the sample volume, F_{down}
- An upward force from below the sample volume, F_{up}
- The downward force of gravity due to the mass of the volume, F_g

If we consider the areas of the top and bottom faces of our volume, which we assume to be equal to A, we can describe the downward force as being due to a pressure, P_{down}, from above while the upward force is due to a pressure, P_{up}, from below:

$$\begin{aligned} F_{down} &= P_{down}.A \\ F_{up} &= P_{up}.A \end{aligned}$$

Through Newton's second law of motion, $Force = Mass.Acceleration$, the definition of density, $Density = \frac{Mass}{Volume}$, and the volume of a prism, $Volume = Area.Height$, the downward force of gravity can first be expressed in terms of the density ρ, and volume V of the sample volume, and then rearranged in terms of A and the height of the sample volume, h:

$$F_g = \rho.g.V = \rho.g.A.h$$

where g is the acceleration due to gravity. Recalling the beginning of this discussion, we stated that because our volume is static any opposing forces must

[5]As with all physical approximations, it is important not to take them too literally, or to accept them as absolute truth. The interested reader is directed to Bohren and Albrecht [1998, p. 35] for a discussion on this subject, and on the practicalities (or rather, impracticalities) of this particular assumption.

be equal and opposite in value. We have defined three such forces, so the sum of their values will be equal to zero:

$$F_{down} + F_g - F_{up} = 0$$

where the sign change is because F_{up} acts in the upward direction, while the other two forces act downwards. Through substitution, it can then be shown that:

$$\begin{aligned} P_{down}.A + \rho.g.A.h - P_{up}.A &= 0 \\ P_{down} - P_{up} &= -\rho.g.h \end{aligned}$$

The expression $P_{down} - P_{up}$ is essentially the change in pressure vertically across the volume so, as the height of the sample volume tends towards zero, we can express the differential change in pressure vertically across the volume with respect to height:

$$dP = -\rho.g.dh$$

This is the *hydrostatic equation* which, through the use of the ideal gas equation, we can express in terms of the pressure and temperature of the sample point, giving:

$$\frac{dP}{P} = -\frac{g}{R_d.T}.dh$$

The solution to this equation may then be found: [Bohren and Albrecht 1998, p. 54]

$$\frac{P}{P_0} = exp\left\{-\int_0^z \frac{g}{R_dT}.dh\right\}$$

where P_0 is the pressure at the base height. However, the integral on the right-hand-side implies knowledge of the exact vertical temperature profile, which is often unavailable. A good approximation can be achieved by using an average temperature value [Rogers and Yau 1989, p. 28].

$$P \approx P_0 exp\left\{-\frac{g}{R_d\overline{T}}(h - h_0)\right\}$$

where $\overline{T}$ is the mean temperature from the base height h_0 to the sample height h. This is known as the *barometric formula* and it provides a reasonable description of how the atmosphere's pressure changes with height.

Following on from this approximation, if the temperature lapse rate is a known quantity, the equation can be simplified into the following: [Auld and Srinivas ca.2008]

$$P \approx P_0 \left(\frac{T}{T_0}\right)^{\frac{g}{LR}}$$

where L is the lapse rate.

In the previous section we stated that a single lapse-rate cannot accurately describe the atmospheric temperature gradient and that the aim is not to use such methods in this work. However, from the hydrostatic and barometric equations if we were to use an arbitrary temperature profile we would either have to approximate its integral or its average temperature, which could be slow to compute at interactive frame-rates. Another solution is to take a further step back from reality and use a simpler approximation: [Bohren and Albrecht 1998, p. 55]

$$P \approx P_0 \exp\left\{-\frac{h}{H}\right\}$$

where P_0 is the base pressure and H is the *scale height* constant, set to 7.29 in this work as defined by Bohren and Albrecht. This equation has the advantage of being independent of temperature, which should simplify the calculations by removing the need to integrate the temperature up to the sample height at the cost of accuracy.

2.3.1.3 Atmospheric Density

The atmospheric temperature and pressure both decrease with height, so the ideal gas equation implies that density must also exhibit a similar trend, which it does. In general, this method of reasoning can be extended to any equation that involves density, for which it should be possible to use the ideal gas law to express it in terms of pressure and temperature instead. As well as potentially simplifying equations this can also be a useful aid to understanding, since it is often difficult to visualise the three-way relationship between pressure, temperature and density, and their implications.

2.3.2 Parcels

Parcels are discrete, traceable, volumes of air [Petterssen 1940]. Andrews [2000, p. 29] likens a parcel to the volume of air contained within a “thin balloon”, which is free to travel through the atmosphere. If the boundary of the parcel is considered not to conduct heat, and if the parcel itself does not generate or absorb heat, then any processes performed on or by the parcel, such as movement, are termed *adiabatic* [Rogers and Yau 1989, p. 6].

Once again, it is important to note that parcels represent a level of abstraction, and approximation, that are just a conceptual mechanism to aid understanding, and to help formulate equations that describe basic behaviour. Reality is far more complex, as a real volume of air will “rapidly mix with its surroundings” [Andrews 2000, p. 29], and would be impossible to trace.

In order to make parcels a useful mathematical tool a number of standard assumptions are made about their properties:

- The air within a parcel is a homogeneous ideal gas and is not considered to mix with the air outside the parcel.
- The actual size of a parcel is not usually specified, though Lawrence [Lawrence ca.2008, p. 3] suggests that, at least for convective clouds, parcels should be thought of “with dimensions of order 10 - 1000 metres”.
- Parcels are considered to move through the atmosphere slowly, in relation to the speed of sound [Lawrence ca.2008, p. 5].
- The pressure of a parcel can be assumed to equalise to that of the ambient atmospheric pressure [Rogers and Yau 1989; Andrews 2000, both p. 29].
- The parcel’s density, volume and temperature are not set in relation to background atmospheric properties, and so are considered properties of the parcel. Since these properties, together with pressure, are related by equations of state, they can be calculated at any point. To achieve this, another property must be included which is related to the parcel’s temperature, and is discussed below.

As a parcel moves through the atmosphere adiabatically its pressure will change, as it equalises to the background pressure. From the ideal gas law, it should be clear that a change in pressure must result in a change in either density, temperature, or both. However, this law by itself cannot be used to determine the parcel’s other two properties when only one is given. Thankfully, there exists a property which is a function of temperature that is conserved during adiabatic processes, and is known as the *potential temperature*[6], usually denoted θ:

$$\theta \equiv \left(\frac{\hat{p}}{p}\right)^{\kappa} T$$

[6]The definition of potential temperature follows on from the definition of an adiabatic process and is derived from the first law of thermodynamics, that energy is conserved, coupled with the ideal gas equation. For further details, the reader is directed to standard texts, such as Rogers and Yau [1989, pp. 6-7], Andrews [2000, p. 28], Bohren and Albrecht [1998, p. 106,157].

where κ is a function of gas constants approximately equal to 0.286. The potential temperature can be thought of as the temperature that a parcel would have if it were to be brought adiabatically to a reference pressure $\hat{p}$.

Thus armed with a means of calculating temperature as well as pressure, the parcel's density and volume can also be calculated should they be required.

2.3.3 Parcel Buoyancy

The term *buoyancy* refers to the resulting movement when gravity acts upon differences in density between a volume and its surroundings. Thus, a volume which is less dense than its surroundings will rise. As previously stated, it is convenient to remove density from calculations and it is possible to do just this with buoyancy. In an ideal gas, which exhibits no pressure difference between the sample volume (i.e. the parcel) and its surroundings, it can be shown [Emanuel 1994, pp. 6-8] that for a unit mass the resulting force due to buoyancy is related to the difference in temperatures:

$$F_B = gB = g\left(\frac{T - T'}{T'}\right) = g\left(\frac{T}{T'} - 1\right) \tag{2.2}$$

Where T is the parcel temperature, in an ambient environment of temperature T' [Rogers and Yau 1989, p. 30].

2.4 Water and Moist Air

So far we have examined the basics of the parcel model, by considering dry parcels moving through a static background atmosphere. In this section we will look at ways of including water parameters into our model, so that we can move towards our goal of using it to describe clouds. We shall look at this in three distinct stages: first we will examine how to categorise water; second, how water can be transferred between categories; and third, how these two aspects are formalised, so that we can use them in our parcel model.

2.4.1 Categories of Water

Water exists in three phases within our atmosphere. These are the gas phase, known as water vapour; the liquid phase, water; and the solid phase, ice. When water in its vapour form changes its phase to liquid, the process is known as *condensation*, and the reverse process is known as *vapourisation* or *evaporation.* Similarly, water changing from its liquid phase to solid is known as *freezing*, and the reverse is *melting*. Water may also change directly from gas to solid, which is known as *deposition*, and the reverse process is *sublimation*.

Water vapour is a colourless gas, so water in our atmosphere is only visible when it is in either the liquid or solid phases. Liquid water particles with a radius of less than about $100\mu m$ [Rogers and Yau 1989, p. 83] have generally so little mass compared to their surface area (and hence ability to be blown about in the wind) that gravity does not play a significant role in their motions. The result is that they remain suspended in the air in the formations that we know as clouds or, if they form close to the ground, fog and mist. The average size of a cloud droplet is generally considered to be about $10\mu m$ [Rogers and Yau 1989, p. 83].

The forms that solid water can take are extremely wide-ranging. Ice crystals grow in what is often believed to be an infinite number of variations, as anyone who has attempted to compare snowflakes can attest to[7]. Like water droplets, very small ice crystals have too little mass for gravity to bring them down to earth (when compared to the drag forces resulting from their sizes), and remain suspended in the air as clouds. If we take a look back at the temperature gradient of the ICAO Standard Atmosphere (Figure 2.9 on page 30), the freezing point generally occurs at an altitude in the mid-troposphere. As such most high level clouds are actually clouds of ice, while mid-level clouds are often made of liquid water at their bases and ice at their tops. An interesting feature of ice-clouds is that if crystals of a similar configuration are present in a relatively thin cloud at high altitude, they can become aligned and result in a variety of optical phenomena due to their refractive properties [Kokhanovsky 2006, pp. 219-223].

2.4.1.1 Precipitation

In general, water that falls out of a cloud in any form is known as *precipitation*. Liquid water droplets that are larger than about $100\mu m$ will usually fall as rain, but droplets larger than about 3.5mm can become unstable and split into smaller droplets[8]. The maximum speed at which a precipitation particle falls is defined as the point at which the gravitational force acting on the particle is balanced by the drag force acting against it as it moves through the atmosphere. The gravitational force is naturally due to the particle's mass, while the drag force is a function of the particle's size and the local atmospheric density.

[7]Magono and Lee [1966] did just this, and provided a detailed classification of snow crystals comprising of some 8 primary categories, presented in a fully illustrated table resulting in 80 entries of unique types.

[8]Houze [1993, p. 80] presents an empirically-based formula developed by Srivastava [Houze 1993 cites Srivastava 1971], in which the probability of a droplet breaking up is given as a function of its size, and results in an exponential increase in likelihood of break-up in droplet sizes beyond 3.5mm.

Although precipitation is a common, and perhaps all too frequent, occurrence on most parts of the planet, it is worth taking a moment to clarify the various terms:

Rain	Falling liquid water of varying droplet sizes. In large cold clouds, this can actually start falling as ice or snow which then melts at lower, warmer, altitudes. Maximum fall velocities for rain drops are around 9m/s at ground-level, but this can be closer to 13m/s where the atmospheric pressure is only 500mb (approximately 5km in altitude). [Rogers and Yau 1989, pp. 124-126] and [Houze 1993, pp. 75-76].
Drizzle	Very fine rain, of radius approximately between 100μm and 250μm [Houze 1993, p. 76].
Snow	Delicate ice-crystals of varying shapes and sizes. Due to their large size relative to their mass, these tend to fall slowly with maximum speeds ranging between 0.3-1.5m/s [Houze 1993, p. 95].
Freezing-Rain	This starts as ice or snow, but as it falls it passes through a warm layer of air and melts. If it then passes through another layer of colder air, just before it hits the ground, it 'supercools' and freezes upon impact.
Sleet	Mixture of frozen and liquid water.
Hail	Roughly spherical ice formations. These can range in size up to over 17cm in diameter[9] and, due to their high density, can fall with speeds of between 10-50m/s [Houze 1993, pp. 95].
Graupel	Snow crystals with additional ice growth. These look like small snowballs, are white in appearance and lighter than hail. They tend to fall with maximum speeds of between 1-3m/s [Houze 1993, p. 95].
Virga	Any precipitation that evaporates before reaching the ground, often forming a new cloud which is usually visible as streaks in the direction the precipitates were falling.

2.4.1.2 Water-Content Representation

A variety of terms exist for describing the level of water present in air. The one most frequently used for parcels is the *mixing ratio*, a unit-less value which

[9]According to the National Geographic News [National Geographic 2003], the largest recorded hailstone fell in Nebraska, in the U.S., in August 2003 and measured 17.8cm in width.

specifies a mass of water present in a given mass of air. A mixing ratio is usually denoted by q (or often w in texts) with subscripts offering further clarification, for example q_v is usually the water vapour mixing ratio. Although technically they are dimensionless, it is common for them to be expressed in terms of a mass per unit mass, such as kg / kg, or g / kg. It can also be useful to express them in terms of pressure and *partial pressures*. A gas is typically made up from molecules of different constituent gasses. According to Dalton's Law [Benenson et al. 2006, p. 732], the total pressure of a gas is the sum of the pressures of its constituents, which are each known as a *partial pressure*.

Examining the case of q_v, by definition:

$$q_v = \frac{m_v}{m_d} = \frac{\rho_v}{\rho_d}$$

Where m_v is the mass of vapour, m_d is the mass of dry air, ρ_v is the density of the vapour, and ρ_d is the density of the dry air. This second formulation is achieved through the general definition of density: $m = V\rho$, where V is the volume occupied, which may be cancelled out of the equation since both masses occupy the same volume. From the ideal gas equation (Equation 2.1 on page 19), we can define the partial pressure, e, of the vapour as:

$$e = \rho_v R_v T$$

Similarly, the dry air pressure may be defined as:

$$p_d = \rho_d R_d T$$

Thus:

$$\begin{aligned} q_v &= \frac{\rho_v}{\rho_d} = \frac{\left(\frac{e}{R_v T}\right)}{\left(\frac{p_d}{R_d T}\right)} \\ &= \frac{R_d e}{R_v p_d} \end{aligned}$$

Where we can evaluate the gas constant ratio as:

$$\frac{R_d}{R_v} \approx 0.622$$

Further, Dalton's law of partial pressures indicates that the total pressure of the atmosphere, p, can be expressed as the sum of the partial pressures of dry air and water vapour:

$$p = p_d + e$$

Thus, we can obtain the equation:[10]

$$q_v = \frac{0.622e}{p - e} \tag{2.3}$$

2.4.2 Water Transfers: Cloud Microphysics

Now that we have empirically defined the various forms that water can take, it should be clear from common observation that water masses can be transferred between those states — such as the melting of ice to become liquid water. There are a number of such transfers, each of which may have a number of different approaches for approximation. Here, we shall examine a number of these transfers, defining equations for some of the simpler ones but looking at them mainly from an empirical point of view. We will formulate mathematical definitions for them later in Section 4.2.4 on page 84.

2.4.2.1 Vapour to Liquid or Ice: The 'Saturation' Myth

Bohren and Albrecht [1998] identify the term 'saturation' as somewhat misleading — in fact they felt it was such a problem that they dedicate several pages to its discussion. In short, they explain that the term 'saturation' erroneously implies that air acts like a sponge, to be filled with water vapour until it can hold no more. They carefully and logically consign this theory to the status of myth and describe the actual phenomenon by starting again from the beginning:

If we consider a body of water, its temperature is essentially the average of its molecules' kinetic (movement) energies. When this body of water interfaces with a volume of air, some of the faster molecules occasionally break free from the attractive forces that hold the liquid together and enter the volume as vapour. This is the process of evaporation and the reverse, where water molecules from the gas collide with and become part of the liquid, is condensation. The key is that both processes occur *continuously* and that they are essentially *independent* from one another. For the most part, the rate of evaporation is dependent on the properties of the liquid and the rate of condensation is dependent on the properties of the gas. Thus, a liquid of higher temperature will result in a higher rate of evaporation, because there are more fast-moving molecules so a greater number will be likely to break free. Similarly, if the gas contains a larger number of water molecules, they are more likely to collide with the liquid. Thus, an increase in water vapour within the gas will increase the rate of condensation.

Part of the terminology confusion occurs because, for the most part, when the terms 'condensation' or 'evaporation' are used, the speaker is referring to the

[10] This derivation was expanded from that presented by Bohren and Albrecht [1998, p. 186], although it can be found in many sources.

overall *net* process. Thus, if evaporation occurs at a faster rate than condensation, there is a *net* evaporation. If, however, the rates of condensation and evaporation are equal, there is no net change. This point of equilibrium is the unfortunately named *saturation* point[11].

Putting this into practice, the ‘saturation’ equilibrium point is fundamental to the development of clouds. Let us consider a parcel with a quantity of water vapour, q_v, and a temperature, T, in a dry atmosphere which, at the position of the parcel, has a pressure p. If the conditions the parcel is in are such that evaporation occurs at a higher rate than condensation, it is reasonable for us to assume that the parcel’s water will remain as vapour. Were any liquid water droplets present, they would quickly evaporate. If, however, the net process is one of condensation, the vapour contained in the parcel may condense into droplets of liquid water and thus form a cloud. Knowing ‘which side’ of the equilibrium point the parcel is at a given time would therefore determine the level of cloud present.

Given the properties of the parcel and its environment, it is possible to calculate the mixing ratio at which equilibrium occurs. This is the *saturation vapour mixing ratio*, q_{vs}, which can be expressed in terms of pressure and partial pressure by applying equation 2.3 to obtain:

$$q_{vs} = \frac{0.622 e_{sl}}{p - e_{sl}}$$

Where e_{sl} is the liquid *saturation vapour pressure*, which can be calculated using an empirical approximation: [Bolton 1980; Rogers and Yau 1989, p. 16]

$$e_{sl} \approx 6.112 \exp\left(\frac{17.67T}{T + 243.5}\right)$$

According to Bolton [1980], this equation is accurate to less than 0.1% between $-35°C \leqslant T \leqslant 35°C$. The units in this equation for saturation vapour pressure are millibars.

Unfortunately, this explanation is not the entire story, because pure water does not easily condense to form new droplets. For this to occur, the amount of water present has to be several times that of the saturation level. In general, when the water level present is higher than the saturation level, it is termed *supersaturation*. In reality, the atmosphere is full of impurities in the form of small particles, around which water finds itself able to condense. Such particles are therefore known as *condensation nuclei* and the process of water forming on them, known as *nucleation*, is a complex subject that is mostly beyond the scope

[11]This explanation is a summarised version of that by Bohren and Albrecht [1998, p. 181] and for a more rigorous and detailed explanation the reader is referred to this text.

of this work[12]. When pure water condenses into droplets, the process is known as *homogeneous nucleation*, while that involving condensation nuclei is known as *heterogeneous nucleation* [Houze 1993, p. 69].

The basic transfer between vapour and ice can be treated in a similar manner to that of liquid. The ice saturation vapour mixing ratio is thus similarly defined as:

$$q_{vsi} = \frac{0.622 e_{si}}{p - e_{si}}$$

The ice saturation vapour pressure e_{si} is then estimated using Buck's empirical method, which is optimised for temperatures between 223.15K to 273.15K (-50°C to 0°C): [Buck 1981]

$$e_{si} \approx 0.6112 \exp\left(\frac{22.452T}{T + 272.55}\right)$$

The reverse processes of evaporation (liquid to gas) and sublimation (solid to gas) are covered later in Section 4.2.4.6 on page 90.

2.4.2.2 The Bergeron Effect:

An important transfer from water to ice is due to a process known as the *Bergeron effect*. This is dependent on the fact that the saturation vapour pressure for liquid is greater than that for ice. Following Storelvmo et al. [2008] there are three possible states for a parcel to be in, with respect to its vapour pressure e_v:

1. $e_v > e_{sl} > e_{si}$, i.e. there is a lot of water vapour. This results in both ice and liquid droplet growth.

2. $e_{sl} > e_v > e_{si}$, i.e. liquid should be evaporating, but ice should be forming. This is the Bergeron effect.

3. $e_{sl} > e_{si} > e_v$, i.e. there is not much water vapour present. This results in both ice and droplet shrinkage.

When the required conditions are satisfied, water is effectively transferred from the liquid to ice categories. In a water model, this could be achieved either implicitly by accurately modelling the transfers between the three categories of water, vapour and ice, or explicitly with a direct formulation of the Bergeron effect.

[12] Rogers and Yau devote an entire chapter to this subject and the interested reader is directed to their work [Rogers and Yau 1989, Chapter 6]

2.4.2.3 Autoconversion

This is the growth of droplets to precipitation size. It is dependent on vapour diffusion (condensation on existing droplets) and coalescence (merging of smaller droplets).

2.4.2.4 Collection

As droplets fall through the cloud, they encounter other droplets and merge with them, 'collecting' them up to form a larger droplet. This is dependent on droplet mass, fall speed, air density, and a 'collection efficiency'.

2.4.2.5 Sedimentation

In our parcel model, this is rain falling into or out of the parcel, to or from other parcels.

2.4.2.6 Aggregation

This is similar to collection but involves ice particles collecting other ice particles. It is strongly dependent on temperature and crystal type.

2.4.2.7 Riming

This is the process of ice crystal enlargement that occurs when ice particles collect liquid water and form larger ice particles. Heavy riming results in graupel.

2.4.2.8 Energy Release and Latent Heat

One important point to note, from the discussion on saturation, is that during the evaporation process the loss of high-energy molecules from the body of water results in a lower average energy within that body. Evaporation is thus a cooling process and, it follows, condensation is a warming process. A net evaporation or condensation will therefore result in a change in temperature of the liquid water.

In general, when water undergoes any phase change, an amount of energy is either released or absorbed. When water vapour condenses it releases an amount of energy L, which is called the *latent heat of vapourisation*[13]. Within a parcel an amount of water, ∂q, changing state from gas to liquid will result in an amount of energy Q being released:

[13]For a discussion on the evils of the term "latent heat", the reader is referred to Bohren and Albrecht [1998, p. 194, 290]

$$Q = L\partial q$$

The property of the parcel that this energy changes is its temperature. For a unit mass, the energy associated with a change in temperature, ∂T, can be expressed as:

$$Q = c_p \partial T$$

where c_p is the *specific heat* of water, literally the amount of energy required to change the temperature by one degree, with different constant values depending on whether the water being heated is liquid, solid or gas. It should be noted that there are two variations of specific heat, depending on whether pressure or volume is kept constant during the process, both variations having different values. Within the parcel model described thus far, it is assumed that a parcel's pressure is equalised to that of its surroundings, so we shall assume that pressure is the constant term.

Returning to the equations, we can now obtain an expression for the change in temperature associated with a phase-change:

$$\begin{aligned} L\partial q &= c_p \partial T \\ \partial T &= \frac{L\partial q}{c_p} \end{aligned} \tag{2.4}$$

2.4.3 Water in Parcels: Continuity Models

So far, our parcel model could only describe a dry volume of air. From the previous section we now know that water may be present in its three phases: vapour, liquid and ice, as well as in various forms of precipitation. We can also describe the quantities of water that are present in terms of mixing ratios. It is therefore time to 'add water' to the parcel mix. To do this, we need to formalise the categories of water and to define a set of equations that describe how water moves from one category to another.

If for a moment we exclude the various processes of precipitation, we can assume that the total water within a parcel will remain constant. As the parcel's properties change while it moves about through the atmosphere this water may undergo phase changes, and water droplets may get larger or smaller. To help keep track of the various forms of water, we can then assign them categories. For instance, water vapour, liquid water droplets, and ice crystals. At its most accurate, an *explicit* model would describe these categories using size distributions, and even shape configurations in the case of ice. However, the computational implications of such a model prohibit its general use, so *bulk* models were developed to discretise the categories to varying degrees of approximation. The set of equations that describe the transfer of water between the categories are known as *water continuity* equations and are based on varying and complex microphysical processes. Different water models take into account different sets of processes and for each process there are different approximations.

2.4.3.1 Bulk Water Continuity Models

Houze [1993, p. 101] builds up a bulk model, starting with a simple two-state scheme. Here, water is present as either vapour, denoted by its mixing ratio q_v, or as 'cloud liquid-water', a term introduced to represent condensed liquid water that forms a cloud, denoted by the mixing ratio q_c. There are no precipitating categories in this model so the total water mixing ratio is conserved, $q_T = q_v + q_c$, and the water-continuity model is then defined by two equations:

$$\begin{aligned} \frac{Dq_v}{Dt} &= -C \\ \frac{Dq_c}{Dt} &= C \end{aligned}$$

where the term C is the result of a function that evaluates the microphysics and acts as either a *source* or a *sink* — that is, it removes water from one category and adds it to another. In this case, the term C "represents the condensation of vapour when $C > 0$ and evaporation when $C < 0$" [Houze 1993, p. 102]. It is typical for water continuity models to be presented in this way, with generalised source and sink terms, since the approximations used for the microphysics vary between different works.

A model that incorporates just two water categories, however, is of limited use. Houze expands his model to include a precipitating rain water category, with mixing ratio q_r, and the equations are expanded to include a number of microphysics sources and sinks:

$$\begin{aligned} \frac{Dq_v}{Dt} &= -C_c + E_c + E_r \\ \frac{Dq_c}{Dt} &= C_c - E_c - A_c - K_c \\ \frac{Dq_r}{Dt} &= A_c + K_c - E_r + F_r \end{aligned}$$

where the sources and sinks are given by:

C is condensation, thus C_c is condensation of cloud water;

E is evaporation, thus E_c is evaporation of cloud water, and E_r is evaporation of rain water;

A_c is autoconversion;

K is collection;

F is sedimentation.

Finally, Houze presents the six-state cold-water scheme described by Lin et al. [1983]. This model describes the changes to the following mixing ratios of a parcel: vapour, cloud water, rain, cloud ice, snow and graupel. The water continuity equations for this are:

$$\frac{Dq_v}{Dt} = (-C_c + E_c + E_r)\delta_4 + S_v$$

$$\frac{Dq_c}{Dt} = (C_c - E_c - A_c - K_c)\delta_4 + S_c$$

$$\frac{Dq_r}{Dt} = (A_c + K_c - E_r + F_r)\delta_4 + S_r$$

$$\frac{Dq_i}{Dt} = S_i$$

$$\frac{Dq_s}{Dt} = F_s + S_s$$

$$\frac{Dq_g}{Dt} = F_g + S_g$$

The S terms represent the total sources and sinks for the cold-water processes, and:

$$\delta_4 = \begin{cases} 0 & \text{if } T < -40°C \\ 1 & \text{otherwise} \end{cases}$$

The mixing ratios are given by:

q_v, the mixing ratio for water vapour;

q_c, the mixing ratio for cloud water — liquid droplets that are too small to fall out of the cloud;

q_r, the mixing ratio for rain water;

q_i, the mixing ratio for ice — solid water crystals that are too small to fall out of the cloud;

q_s, the mixing ratio for snow;

q_g, the mixing ratio for graupel.

The format presented here, which is from Houze [1993, p. 105], is highly abstracted from the microphysics of the sources and sink factors. The original work by Lin et al. [1983] includes full explanations and formulas for their 27 source/sink factors. We will revisit water continuity models later in Section 4.2.4 on page 84, where simulations are discussed and developed.

2.4.4 Moist Parcel Buoyancy

The previously stated buoyancy equation (Equation 2.2 on page 36) holds true under dry conditions, but gravity will act against any liquid or solid water that is present. While, in the case of small particles, this is not large enough to make them fall from the cloud, it does have an effect on the buoyancy of the parcel. In such a situation, the following expanded buoyancy equation can be used: [Rogers and Yau 1989, p. 50]

$$B = \frac{T}{T'} - (1 + q) \tag{2.5}$$

Where q is the parcel's total mixing ratio of all condensates, which is essentially all water categories except vapour.

2.5 Atmospheric Stability

According to the buoyancy equation, a dry parcel that is warmer than the ambient or background temperature will rise. As the parcel rises the environment's pressure decreases and thus so does its own. This results in a decrease in the parcel's actual temperature and the rate at which this occurs as it rises adiabatically is known as the *dry adiabatic lapse rate* (DALR). Andrews [2000] gives this value at approximately $9.8\,K.km^{-1}$ and it is important to note that this corresponds with a constant potential temperature. If the rate at which the background temperature decreases (its *lapse rate*) is greater than the DALR, the difference between the parcel's actual temperature and the background temperature will increase as the parcel rises, resulting in an increase in buoyancy, and the atmosphere is said to be *unstable*. If the atmospheric lapse rate is identical to the DALR there is no buoyancy force and a displaced parcel will remain in its new location, in which case the atmosphere is said to be *neutral*. Finally, if the atmospheric lapse rate is less than the DALR a rising parcel will exhibit a gradual decrease in buoyancy until it reaches an equilibrium altitude at which the background atmospheric temperature equals its own actual temperature. If the parcel moves above this point, the buoyancy force will act in the opposite direction and attempt to push it down to its equilibrium altitude. Such an atmosphere is termed *stable*.

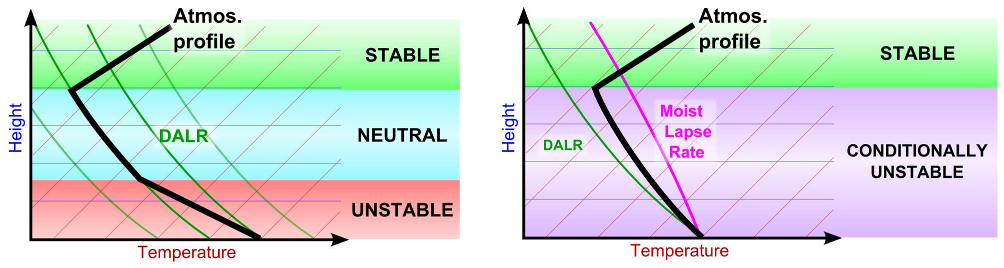


Figure 2.10: The three states of dry atmospheric stability and the additional state for moist parcels. A rising parcel will increase its rate of ascent through an *unstable* region. A displaced parcel in a *neutral* region will remain in its new location. A parcel in a *stable* region will rise or fall to find a level of equilibrium. A *conditionally unstable* region is *stable* for a dry parcel and *unstable* for a moist parcel.

However, a fourth state exists for water-laden parcels. When water condenses latent heat is released and the parcel's temperature increases, so it no longer follows the DALR. Instead it follows a new lapse rate called the *saturated adiabatic* lapse rate, or the *pseudoadiabatic* lapse rate. This rate is not constant, but is less than the DALR and so creates another stability state where the atmospheric lapse rate is less than the DALR (and thus stable for a dry parcel) but greater than the pseudoadiabatic lapse rate (and thus unstable for a moist parcel). Such an atmospheric region is termed *conditionally unstable*. These rates are shown graphically in Figure 2.10.

2.5.1 Analysis Tools

The tool that is used here is the *Skew-T log(P)* graph (commonly referred to as just a "skew-T") [Air Weather Service 1990]. This is one of a family of similar thermodynamic charts commonly used by meteorologists to plot an atmospheric profile to show how temperature varies with height. These charts vary only by the method of plotting their coordinate axes of temperature and pressure. In the skew-T, temperature is 'skewed' at 45 degrees and the logarithm of pressure is used (hence the diagram's name) see Figure 2.11. Because the pressure of the atmosphere decreases exponentially with height, to a reasonable approximation, height can be used in place of pressure as one of the primary axes. This work uses a temperature-height axis graph, assuming that any potential user is likely to be more comfortable with these properties.

In addition to the base axes and often running nearly perpendicular to the temperature are curves of constant potential temperature (in green). The DALR represents a constant potential temperature so these lines can be used to quickly

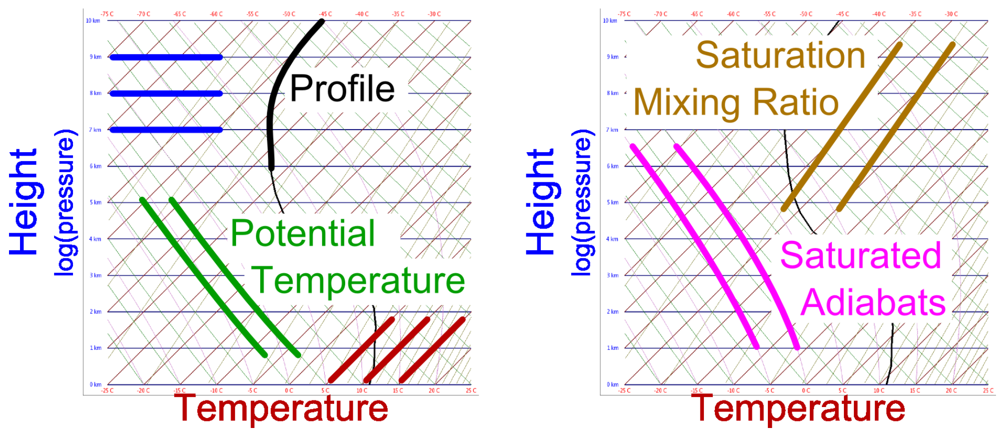


Figure 2.11: The axes of the Skew-T log(p) graph, and a sample profile. The background graphs are screenshots from the prototype tool developed as part of this work.

identify whether a profile segment is stable or unstable. We can also see how high a dry parcel of a given potential temperature is likely to rise, by starting at ground level and following its potential temperature line up to the point where it crosses the atmospheric profile line. It is important to note, however, that a rising parcel which encounters a stable portion of atmosphere can still continue to rise due to its momentum, but the buoyancy force will attempt to return it to the altitude where it crossed the atmospheric profile line.

Running nearly parallel to the temperature axis are curves of constant saturation mixing ratio (in brown). This is traditionally used to plot the dew-point associated with the temperature profile, which gives a measure of how much moisture is present in the atmosphere. If we consider a parcel rising from ground level, following a potential temperature line we see that it crosses through the saturation mixing ratio lines. If we choose an altitude at which we want a cloud to form, we can examine one of the saturation mixing ratio lines to find the vapour mixing ratio that our parcel would need to contain in order for it to become saturated (i.e. a cloud to start forming) at this height. This point is known as the *lifted condensation level* (LCL). As the parcel rises beyond this height, the saturation mixing ratio drops further and more water condenses. Condensation releases latent heat and warms the parcel, so we must stop following a line of constant potential temperature and instead follow a *saturated* or *moist adiabat* line (in pink). As the parcel continues to rise more water condenses and it may even start precipitating.

The skew-T is a versatile analysis tool with additional features that can be identified, but the brief description here is sufficient for the purposes of this work. We will consider ways of using this tool later in Section 4.2.3.1 on page 82.

2.6 Empirical Formation Models

From the review of clouds in the introduction, we can conclude that there are three main types of cloud form, which actually correspond to Howard's original description: *cumuliform*, *stratiform*, and *cirriform*. As might be assumed, their visual appearances do correspond to differences in formation and internal dynamics. In this section we shall look at each of these in turn, and relate them to the parcel model.

2.6.1 Convection and Thermals: Cumuliform Clouds

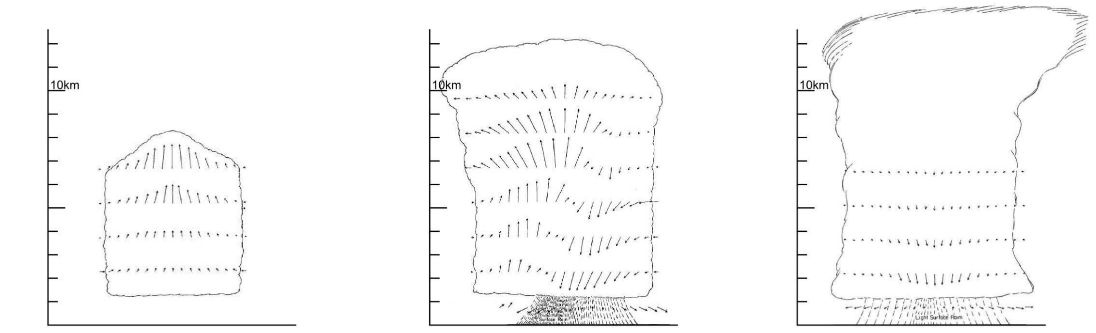
Our parcel model corresponds extremely well to cumuliform clouds, and can be applied to these directly. As previously implied, the first stage of the circulation process assumes that parcels near the ground level are heated by some means. Then, as has been described, the heated parcels will rise through a gradient-temperature environment in an attempt to reach their level of neutral buoyancy. This type of heat transfer, where the heat itself is responsible for its own motion or 'advection', is known as *free convection* [Bohren and Albrecht 1998, p. 354].

One of the difficulties that held back early development of cloud models was the lack of good quality observational data. Early observational data was often anecdotal, with glider-pilots being the main source. The development and improvement of aircraft, radar, satellite imagery and even time-lapse photography changed this, and models can now be tested to a reasonable degree of accuracy against their real-life counterparts. One of the first serious, scientific, studies of clouds was the 'Thunderstorm Project' [Byers and Braham 1948], performed just after World War II, which used aircraft to take a number of basic measurements of cumulonimbus clouds.

2.6.1.1 Cumulonimbus Dynamics

A great deal of effort has been expended over the years in attempts at understanding this form of convective cloud, driven partly by the need to provide warning of their potentially devastating effects.

In the Thunderstorm Project, aircraft fitted with recording instrumentation literally flew through thunderstorms to collect data. The analysis was presented by Byers and Braham [1948], and showed how each thunderstorm was found to consist of a number of mostly independent convective *cells*. Each "thunderstorm cell" was then described as having a life-cycle of three main stages: a growing cumulus stage, a mature stage, and a dissipating stage (see Figure 2.12). It was found that the lifetime of a cell was typically between one and three hours. A



(a) Growing cumulus stage (b) Mature stage (c) Dissipating stage

Figure 2.12: The life-cycle of a storm cell, as described by Byers and Braham [1948]. (Diagrams taken from their paper and adapted)

thunderstorm was described as starting as a single cumulus and, for a few minutes, would consist of just one cell.

The first stage of the cell is characterised with the formation and growth of an *updraught*. This process starts with a cumulus cloud although, of course, Byers and Braham note that "only a small number of cumulus clouds actually build into mature thunderstorm cells". They go on to describe how two or three cumulus clouds may "grow together into one cell", usually with a diameter of 1.5-8km, up to a height of about 4.5km. The main feature of this first stage is the updraught, which is a large upward motion within the cloud "prevailing throughout the entire cell and balanced by gentle subsidence in the environment". The strength of this updraught was found to vary anywhere from between one metre-per-second in a small or weak cell, to nearly thirty metres-per-second in "large well-developed cells". It was also observed that the updraught at ground level exhibited a degree of horizontal convergence of the winds towards the centre of the cell, on a scale equal to that of the cell's radius.

One important point to note is that the lapse rate of a parcel in the updraught was found to be higher (i.e. decreasing at a greater rate) than the moist adiabatic rate. This can be explained by modifying our parcel definition and including another physical process known as *entrainment*, in which a rising parcel mixes with its surroundings. If a warm parcel gradually incorporates atmospheric air that is cooler, and possibly dryer, then it will naturally cool at a faster rate than would be described by the moist adiabatic lapse rate.

The mature stage sees the development of a downdraught, with heavy rain or other precipitation that reaches the ground. Byers and Braham observed that when a cell reaches this stage it will usually have a vertical extent of up to 7.5-9km, and that a downdraught will have formed "in the very region where

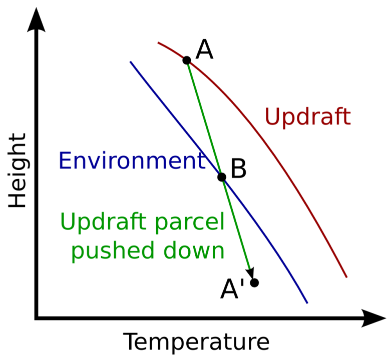


Figure 2.13: Thunderstorm temperature profiles (lapse-rates) in downdraught formation. If a parcel from the updraught at position A is moved downward, its temperature will follow the line A to A'. When this parcel falls below the environmental temperature at point B, it will continue to sink. Adapted from Byers and Braham (Fig. 22) [1948]

updraughts had previously existed". The authors attributed the formation of this downdraught to falling rain, and describe it with reference to the lapse rates within the cloud and the surrounding environment.

A typical parcel in a thunderstorm updraught will be moister and warmer than its environment, despite entrainment, and may show a temperature profile such as the red curve in Figure 2.13, when compared to a typical unstable environmental lapse rate, shown in blue. If such a moist parcel were to be *forced downward* from its position aloft, traditional parcel theory states that it would follow the moist-adiabatic curve in its descent, shown by the green line. At the point, B, where this line crosses below the environmental temperature, the buoyancy of the parcel would become negative since it is suddenly at a lower temperature than its surroundings. Byers and Braham identified that rain can drag air downwards [Byers and Braham 1948 cite Kaplan 1943], and attributed this as the process which triggers downdraughts. Furthermore, if the downward moving parcel entrains more cold air, the process can actually be accelerated.

The downdraughts were observed as starting at a height of around 4.5km, developing up to 7.5km, with velocities varying from a metre per second to over 12 metres per second. It was estimated that rain was required to pull the air down by about 1km before it would start to sink of its own accord. A downdraught was never observed to extend all the way to the top of the cell, which the authors attributed to the lack of sufficient water to form the downdraught-triggering precipitation.

The dissipating stage is characterised by the weakening first of the updraught, and eventually of the downdraught before the final dissipation of the cell. The

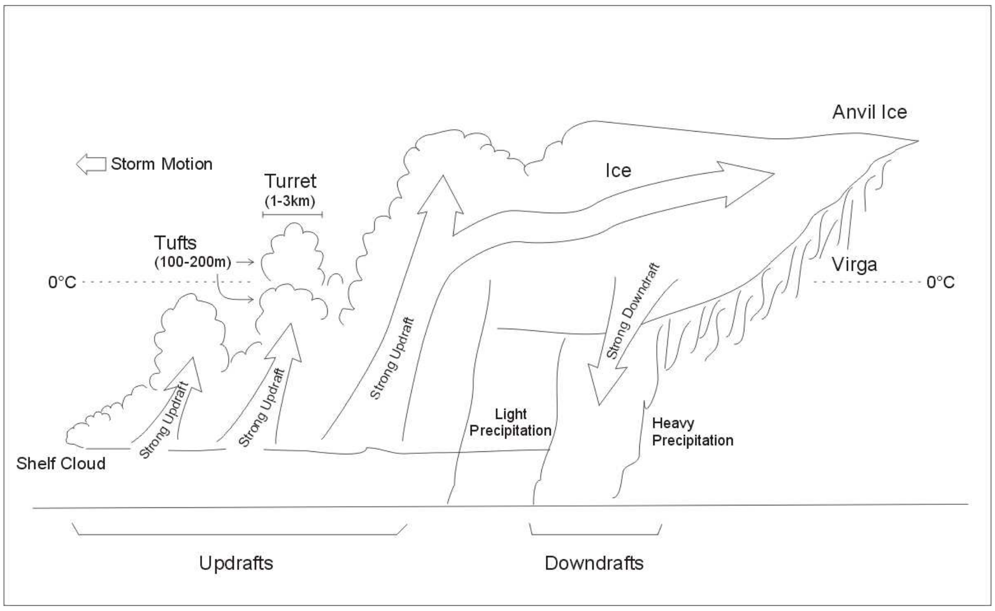


Figure 2.14: Empirical model of a multi-cellular cumulonimbus storm cloud formation. Adapted from Houze (Fig 8.1) [1993, p. 270]

downdraught weakens from less than 6 metres per second to almost zero, and cloud dissipation was observed to occur first at the lower levels.

In general, most large thunderstorms were found to consist of a number of these cells, at various stages of their life-cycles (see Figure 2.14). However, there exists another class of thunderstorm which has been heavily studied, and is much feared in some parts of the world. This is the *supercell*. About the same size as a multicellular storm, these are characterised instead by a single updraught-downdraught pair of a much larger, and more violent, scale. Supercells are associated with gale-force winds, lightning, devastating hail, and are prone to exhibit vortex motions that result in tornadoes. [Houze 1993, p. 278]

2.6.2 Convection and Layers: Stratiform Clouds

These types are generally non-precipitating, with low water contents (typically below 1g/kg). They do not usually exhibit the strong vertical motions commonly associated with cumuliform clouds, instead vertical velocity components are typically between 1 and 10cm/s. With the obvious exception of nimbostratus, stratiform clouds are not usually deep in their vertical extent and are typically under one kilometre, though they can extend to several kilometres. Horizontally, however, they can vary widely and can extend for hundreds of kilometres. Aside from their formation as a result of other cloud remnants, a factor which can be responsible for the formation of most cloud types, there are three main methods of stratiform cloud formation: [Houze 1993, Chapter 5]

The first method of formation involves the cooling of more-or-less static warm air. Here, a volume of warm moist air in close proximity to a cool surface is slowly cooled. As its temperature falls, it can become saturated and the moisture it contains condenses to form a fairly uniform stratus cloud layer. This type of formation naturally occurs at low levels, and can also result in fog.

The second method of formation is due to a *cloud-topped mixed layer*, and in many ways is similar to cumuliform formation, but on a much larger scale. Here, a volume of air is heated from below, for instance by the sea. This causes a large number of convective plumes to rise, which mixes the air. If conditions are favourable, clouds can form at the very tops of the plumes, which appear as the small elements of stratocumulus. If the air continues to mix, especially at the level of the clouds, the individual elements can become merged into a stratus layer. As the clouds age, they dissipate and break up and, although the exact processes are poorly understood [Houze 1993, p. 157], they are often observed to result in cumulus or even small cumulonimbus clouds.

The third method of formation occurs within layers of air high in the atmosphere, where a layer of unstable air is surrounded by stable layers below and above. Warm, moist air from the bottom of the central layer rises convectively, and forms cloud elements. The key here is that solar radiation destabilises the clouds and the middle layer becomes turbulent, thereby mixing the cloud elements which can merge into a stratus layer.

2.6.3 Ice and Snow: Cirriform Clouds

Cirriform clouds occur high in the troposphere, where temperatures can range from between -20 and -85°C. It is typical to find high wind shear at these altitudes, and as such cirriform cloud elements can become stretched. Further, because the clouds consist of ice, they are long-lived and their streaks can extend far across the sky. The reason for their longevity stems from the fact that the saturation vapour pressure for ice is lower than that of water [Rogers and Yau 1989, p. 16]. From Equation 2.3 on page 40, it can be seen that this results in the saturation mixing ratio for ice being relatively low, so a parcel is ‘easy to saturate’ in terms of ice. This means that sublimation will occur at a relatively low rate as a parcel containing ice travels through the atmosphere, resulting in longer-lived ice streaks.

Aircraft observation has indicated that water quantities of between 0.01-0.1 g/kg are typical in cirriform clouds, and that particle sizes range from between 50 and 1000μm. Vertical air motions are typically very slow, typically 0.1-0.2m/s, but can be up to 2m/s in some cirriform types. It should also be noted that this

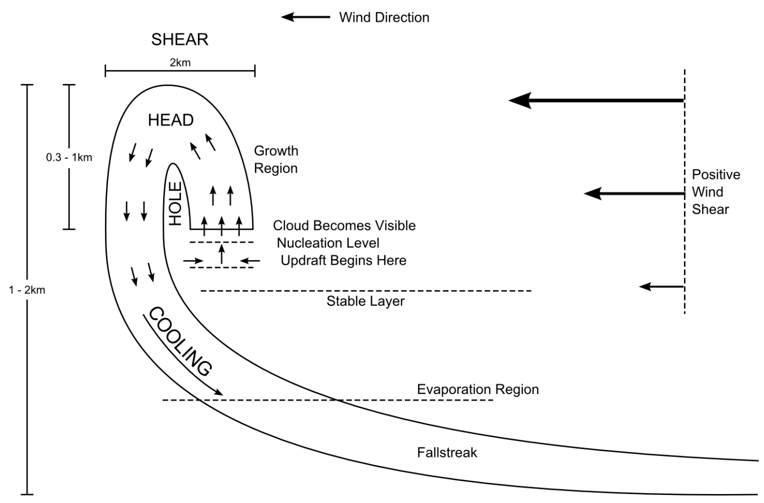


Figure 2.15: Empirical model of a cirrus element, adapted from Houze [1993, p. 177]

category of cloud can be extended to the very tops of the deep vertical clouds, cumulonimbus and nimbostratus.

Typically, then, cirriform clouds consist of a single or a number of cloud elements. These elemental units consist of a dense, convective head, attached to a "long fibrous tail (or *fallstreak*) of falling snow" [Houze 1993, p. 175]. (See Figure 2.15).

The formation of cirriform elements is similar to the third method of formation that was described for stratiform clouds. Again, it occurs in a volume of unstable air, with stable layers above and below, but here the middle layer is much less turbulent. Warm, if such a word can be used to describe an ice-laden parcel, moist air from the bottom of the central layer rises with convection and forms the head of the cloud. Vertical air motions within this head are typically of the scale of 1m/s. As the cloud starts precipitating, a downdraught forms and the characteristic fallstreaks are observed. As this downdraught penetrates the lower stable layer, it is possible that the disruptions can trigger new elements to rise and form additional cirriform clouds. [Houze 1993, Chapter 5]

2.7 Fluid Simulation for Clouds

The atmosphere is considered to behave as a fluid and as such its motions can be described by fluid dynamics. This is basically a system of equations that determine how the parcels move through the atmosphere. As the parcels' properties change, they exert buoyancy forces which are fed back into the fluid solution which must resolve these to produce a smooth motion.

Fluid dynamics is a complex topic and is not the main focus of this thesis. However, it is still important and a full investigation and explanation of the techniques used for this research may be found in Appendix A.

Chapter 3

Related Work

In this chapter we shall look at a variety of techniques and approaches that have previously been employed for the simulation of clouds and other weather effects. In general, the problem can be split into two aspects: simulation and rendering. Many works have concentrated on one over the other, as has this thesis. Here, we are primarily interested in simulation work, so our focus will be on these aspects. Approaches to this topic generally take the form of either physically or procedurally based, and we shall examine each in turn, but with emphasis on physically-based simulations since they are of most relevance to this work.

Due to the large body of related work that has been undertaken over the years, the goal of this chapter is primarily to briefly identify a broad sample, and to direct attention to areas of interest and relevance. We will start by briefly touching on meteorological works, before focusing on those presented within the computer graphics community.

3.1 Meteorological Approaches

The development of computer hardware allowed meteorologists to create 'numerical weather prediction models', which simulate the evolving atmosphere. This also meant that meteorologists could easily test their theories by running a simulation and comparing the results with those observed from real clouds. Modern meteorological simulations are often global in scale, and the grid sizes they employ are usually very coarse in resolution — usually many kilometres in size. The shape and form of individual cloud elements are therefore not often an output of meteorological simulations, so the goals of this class of simulation do not usually align with those of computer graphics. There is, however, a great deal of interest in storm-scale simulations within meteorology, in particular with the ability to predict the devastating effects of tornadic formations. These types of simulation result in individual cloud descriptions, and computer graphics techniques can be used to render them (as will be investigated in Section 3.2.4 on page 67). The simulations themselves are essentially several generations of development on from the techniques presented in this report, and usually require supercomputing-class hardware to run them.

Meteorological centres typically operate a range of models to help predict different weather and climate features. In the UK, the Meteorological Office has developed a flexible ‘Unified Model’ which is configurable for different applications such as global or local, short-term or long-term [Met Office ca.2013]. The European Centre for Medium-range Weather Forecasts (ECMWF) maintains an Integrated Forecasting System which is deterministic, global in scale and able to predict weather up to 10 days in advance [ECMWF ca.2013]. In the US, the National Oceanic and Atmospheric Administration (NOAA) offers many models including the Global Forecast System (GFS) which operates at a grid resolution of 28km per cell and makes predictions up to 16 days in advance [NOAA ca.2013]. These models represent the state-of-the-art in numerical weather prediction and they are deployed on some of the most powerful computers in the world: the Met Office’s IBM Power 775 was ranked 43rd in 2012 while the ECMWF operates a pair of identical IBM Power 775 systems which were ranked 38th and 37th [Meuer et al. 2012].

Another simulation model is ARPS[1], the Advanced Regional Prediction System which is freely available and runs on a wide range of hardware (including PCs). The system uses an Eulerian (grid-based, as detailed in Appendix A) approach, and is capable of working at various different resolutions. Its fluid-dynamics solution uses the compressible form of the equations (we only examine the incompressible form in Appendix A), and tracks potential temperature and pressure. Like most meteorological simulations, the spatial coordinate system is one that is tangent to the earth’s surface, in this case employing a “curvilinear coordinate system” [Xue et al. 2000, p. 164] that vertically stretches the grid non-linearly, so that it has a higher-resolution closer to the surface — which also allows the system to better account for surface height differences. The fluid dynamics model itself also includes a range of sub-grid methods for approximating and preserving turbulence. The water continuity model includes six states, for: vapour, cloud, rain, ice, snow and hail, and also includes approximations for physical processes involving interactions between the land and the atmosphere.

The very nature of and demand for increasingly complex weather prediction systems mean it is unlikely in the foreseeable future that such methods are directly

[1]Details of the simulation system may be found in the publications by Xue et al. [2000; 2001] while other details, including the program itself, are available from the ARPS website: http://www.caps.ou.edu/ARPS/

transferable to computer graphics. In fact, there are a number of fundamental reasons why this is not necessarily a good idea in the first place. Firstly, the sole task of meteorological simulations are the simulations themselves, and all computer resources are dedicated to this task. In computer graphics, especially the realm of computer games, weather simulation is never going to be the 'sole task', and most likely not even one of high-priority in the competition for system resources. Secondly, computer graphics is a creative subject, and in most cases its output is the result of careful and skilful guidance by talented artists. Physically-based systems, by their very nature, are under the sole control of the physical systems they emulate. The only means of influencing such a system is thus by adjusting the parameters of the simulation, or the input data, both of which can be highly unintuitive and the results are usually unpredictable. This is not conducive to artistic direction. There are exceptions, however, where such simulations are still useful — a flight simulator, for instance, is essentially a physically-based system and could be enhanced by accurate weather simulation. Let us now look, then, at how clouds have been addressed within the area of computer graphics.

3.2 Computer Graphics Approaches

One of the primary factors within computer graphics is how long an effect takes to simulate and render. Perhaps the least computationally-expensive solution to produce clouds within 'computer graphics' is simply for an artist to draw them, or for a photograph of a real sky to be taken. The resulting image texture can then be projected onto the inside of a simple geometric primitive, such as a cube or dome (also known as a "sky-box" or "sky-dome" when used for this purpose) [Bell 1998]. This technique is still popular within the games industry, although it does exhibit one obvious flaw: unless the artist produces a whole series of textures or a video of a real sky is taken (both of which are unlikely in a game scenario due to the high memory costs incurred), there will be no animation or evolution of the cloud-scape. Ways around this problem include slowly scrolling the texture across the sky, which could involve a decoupling of the clear sky colour from the cloud layer [Anderson and McLoughlin 2007]. An evolution on this is to use a number of cloud layers, which move at different speeds to emulate clouds at different heights (this is the parallax effect). However, these techniques will still only mimic static clouds, and make them suitable only for short duration shots or very calm conditions at best.

The full simulation of clouds are generally tackled from one of two distinct angles: by simulating them using a physically-based system, or emulating them using procedural techniques. The two approaches are not mutually exclusive, however, and the advantages of both can be combined – for example, Riley et al. [2003] used procedural noise to make physically-based simulation data more visually detailed and appealing.

3.2.1 Previous Physically-Based Simulation Work

Of most relevance to this work are physically-based techniques, of which there are surprisingly few:

The first attempt at using a physically-based cloud simulation for computer graphics was by Kajiya and Herzen [1984]. Their technique used two states of water, with mixing ratios for vapour and cloud liquid. They modelled the potential temperature and included heating due to the latent heat of vapourisation. To account for water continuity, they approximated the saturation mixing ratio as an exponential function of height. Their work was more concerned with rendering techniques, however, and they did not provide full details of their atmospheric motion simulation. Although their technique was massively constrained by the computing power available at the time, it could crudely simulate an evolving cumulus cloud.

Overby [2002] and Overby et al. [2002a; 2002b] used Stam's fluid dynamics method [Stam 1999] (see Appendix A) and a very similar approach to that of Kajiya and Herzen, with the major difference being that the system could run at interactive speeds on contemporary computer hardware. Although the authors claimed their new work could simulate both cumulus and stratus clouds, the simulation of stratus essentially bypassed the formation step, by assuming that the stratus cloud was carried into the simulation domain from an external source by a "high-velocity cross-wind" [Overby 2002, p. 48]. The authors also introduced an additional term into their fluid solution, to account for the expansion of a parcel as it rises through an atmosphere of decreasing density. While Harris et al. [2003] discount this as unrealistic because the fluid solver should handle such effects, Overby et al. did not use a model that included atmospheric density reduction with height. Further, it could be argued that such a factor may be necessary to account for entrainment effects that take place at a scale smaller than the fluid solver's grid resolution. However, they did also introduce a technique for enhancing the momentum of rising parcels, based on the belief that their system failed to properly conserve momentum.

Miyazaki et al. [2002] presented a cumuliform simulation technique, again coupled with Stam's fluid dynamics technique. They did, however, address the issue of numerical dissipation in the advection step (explained in Appendix A, A.4.3 on page 152) with the addition of the *vorticity confinement* technique introduced to computer graphics by Fedkiw et al. [2001]. In this method, the vorticity of cells is estimated and an additional body force is computed to conserve rotational motions [Bridson 2008, p. 131]. Their system again implemented the fundamentals of cloud physics, and included water in the forms of vapour and liquid cloud water, as well as simulating the effects of latent heating and cooling. Typical results of their method achieved 5 sec/frame on grid resolutions of $150 \times 120 \times 50$ on contemporary hardware (Pentium III 1GHz).

The work by Harris [2003] and Harris et al. [2003] introduced the use of graphics hardware to the problem of cloud simulation. Their work essentially solved the same sets of equations as the previous works, with a two-state water continuity model including thermodynamic implications, and Stam's stable fluid solution. However, in this work the entire simulation, including fluid-dynamics solution, was executed on the GPU. The main advantage of this technique is that the GPU is a powerful parallel processor, and as such is generally well-suited to this type of problem. In addition, the simulation update rate was decoupled from the rendering process and the cost of the simulation was spread over a number of frames. Unfortunately, the authors did not take into account the differences between one simulation step and the next, which resulted in a visual 'popping' whenever a simulation update occurred. Their method achieved simulation update rates of 5.7 frames/sec on contemporary hardware (a GeForce 5900 Ultra).

As previously mentioned, one of the major issues with physically-based techniques in general is that they are difficult to control, a factor which is especially important from an artistic point of view. The work by Dobashi et al. [2008] attempts to address this by allowing an artist to define a desired silhouette, or "contour line", that the simulation adjusts itself to fill with cumuliform clouds. Their innovative technique uses Eulerian-based fluid dynamics and cloud control is achieved through the adjustment of latent heat and water vapour within the simulation. As a cloud element approaches its target position, the amount of heating due to water vapour condensing into liquid water is adjusted to influence the buoyancy of the element. This controls the level at which it achieves neutral buoyancy, and stops rising. Similarly, if an element is unlikely to reach its target, additional water vapour is added. Although the level of control is limited to a two-dimensional silhouette, it is likely that the technique could be extended to use a three-dimensional surface as its target, and it would be interesting to see if this could then be animated.

3.2.2 Procedurally-Based Techniques

Due to the computational expense of physically-based methods, procedural techniques have always played a leading role in computer graphics, in both 2D and 3D representations. Further, as previously mentioned, several layers of 2D textures may also be used in a "2.5D" technique.

Procedural methods are particularly good at adding high-frequency detail, which makes them well suited to complement a physically-based solution, in which high-frequency details are typically too computationally expensive to simulate. Since the addition of such high-frequency noise is commonly applied at rendering time, the distinction between simulation and rendering stages can become blurred. In general, however, we argue here that procedural techniques are not particularly well suited for the production of 'low-frequency' cloud descriptions, especially those that evolve with time, simply due to the extreme level of artist input that would be required.

3.2.2.1 2D and 2.5D

Numerous techniques exist for the production of 2D textures that are suitable for the emulation of clouds [Ebert et al. 2003]. In particular, Perlin noise [1985] has always been a popular technique, while pure fractals have also been used [Nishita et al. 1993].

Once 2D textures have been created, they can be used in a number of ways. Again, the simplest approach is their direct application to a sky-box or sky-dome, and they can also be layered, with multiple scales of noise moving at different speeds to emulate an evolving cloud-scape [Pallister 2001; Elias 1998]. Generally speaking, most techniques using 2D textures will also employ a more sophisticated rendering technique than pure texture mapping, such that the texture itself is no longer the final result but merely a parameter to some rendering equation. This means that the geometry on which the texture is placed can be relatively simple. For example, the use of a plane, or multiple planes, on which the textures may be applied and rendered [AMD-ATI 2002; Bouthors et al. 2006][2].

Alternatively, individual sprites or billboards could be used for texture placement, although this technique is perhaps best suited for the application of artist-generated images. Such a technique was used by Wang [2004], who used arrangements of billboards to approximate clouds. Larger formations could be made from multiple billboards, and dynamic weather was possible to a degree. However, the system was designed without any automatic drive for the placement and formation of clouds, so a high level of artistic control was required. The system was fast and efficient, however, and was successfully deployed in Microsoft Flight Simulator 2004 [Microsoft ca.2013]. A further implementation based on this technique appears in the commercial game engine CryEngine 2 by CryTek [Wenzel 2006]. Other billboard approaches have also been employed [Heinzlreiter et al. 2002].

Further, 2D textures can be applied to 3D geometric primitives, such as the

[2]The work by Bouthors et al. [2006] used textured quads when their clouds were viewed from below but, when viewed from above, they constructed more complex geometry from a height-field texture.

work by Gardner [1985] who applied textures built from multiple sine-based functions to ellipsoids to modulate their transparency, and which has since been implemented at interactive speeds [Elinas and Stürzlinger 2000].

3.2.2.2 3D and General Volume Rendering

Since clouds themselves are volumetric by nature, it perhaps makes sense to use full 3D techniques to emulate them. These can generally be divided into surface-based or volume-based methods, although the difference can sometimes purely be a change in rendering technique. Since clouds are themselves volumetric with no distinct surface, there is often a preference for such methods. However, it should be remembered that surfaces can still be useful if their transparencies are carefully modulated [Gardner 1985].

A variety of general purpose volume techniques are available which are suitable for use in cloud systems. The first stage is to determine some form of volumetric description. This can generally take the form of either pre-generated 3D textures, which represent density samples; or some form of implicit functional description, which must be evaluated to retrieve a sample. Methods of generating 3D textures could be regarded as noise-based (such as Perlin noise), or purely procedural, including fractal-based (such as fractional Brownian motion). Functional descriptions include metaballs, hypertextures, and other implicit surfaces. Hybrid techniques also exist, such as perturbing a 3D texture with a procedural offset (which may itself be stored in another 3D texture) [Engel et al. 2006]. The possibilities are essentially only limited by human imagination and the computational resources available. For more information on procedural techniques, the reader is referred to Ebert et al. [2003].

Once some form of volumetric data has been determined, a number of options are available to render it, most of which are suitable for any form of volume data — i.e. those physically or procedurally generated. The data can be converted into a surface for rendering, and polygonal surfaces may also be subjected to geometry reduction techniques for continuous level-of-detail effects [Roettger and Ertl 2003]. Graphics hardware can also be used, and Nvidia [2007] demonstrated a metaball isosurface technique that works on the GPU. Volumetric data can also be rendered using ray-marching style techniques, which are usually achieved in real-time applications with the creation of simple geometric primitives (such as planes) that sample a 'slice' through the volume. Vane [2004] used 3D Perlin noise textures with such a slicing algorithm to approximate clouds. Alternatively, volumes can be rendered using hybrid methods [Co et al. 2003]. For more information on volume methods, the reader is directed to Engel et al. [2006], where a wide variety of techniques and issues are presented and discussed.

A good selection of volumetric techniques have already been used for clouds.

The work by Stam [1991] used a stochastic (random noise-based) approach, while metaballs and 3D fractals have also been successfully used for clouds [Nishita et al. 1996]. In general most 3D approaches to cloud simulation split the problem into two, with a 'high-level' technique to determine the low-frequency general cloud shapes, and a 'low-level' technique for the high-frequency noise and turbulence. Often, the high-level technique is little more than an interface to allow for artistic control, which can lead to an overly heavy burden on artists.

The work by Schpok et al. [2003] followed this two-tiered approach. Their high-level method involved volume implicits, defined by implicit functions that could be controlled by the user of their interactive 'cloud-building' system. In this, the functions were presented in the form of ellipsoidal primitives which the artist could move around and merge together. For their low-level technique, a 3D texture containing periodic noise data (re-used in several octaves) is essentially subtracted from the results of the high-level implicit volume. Their system made heavy use of the GPU and could provide results at interactive rates of between 5 and 30 frames/sec on GeForce 4600 hardware, although they make no mention of how well their technique could scale. Their work was, however, successfully employed in the animated film Valiant [Venere 2005].

3.2.3 Additional Cloud-Simulation Work

By their own admission, the work by Dobashi et al. [1998a; 2000] lies between the realms of physically-based and procedural (or heuristic as they refer to it). Based on an earlier method [Dobashi et al. 1998a cite Nagel and Raschke 1992] they used a cellular-automata based technique to simulate an evolving cloud system, using physically-inspired rules for the transfer of water properties between cells in a volumetric grid. Their system was essentially binary based, and a cell could result in a cloud element being either present or not. The simulation output was then subjected to a smoothing operation and, when combined with an advanced rendering technique, their results were quite convincing, if not interactive — typical times per frame ranged from 10 to 30 seconds on a dual Pentium III 500MHz workstation. Their approach has since been extended [Miyazaki et al. 2001; Liao et al. 2004], while others have also used coupled-map-lattices (a direct extension of cellular-automata) for similar problems [Harris et al. 2002]. The cellular-automata method of Dobashi et al. [1998a] has also been applied to drive a particle system [Brickman et al. 2007a; Brickman et al. 2007b].

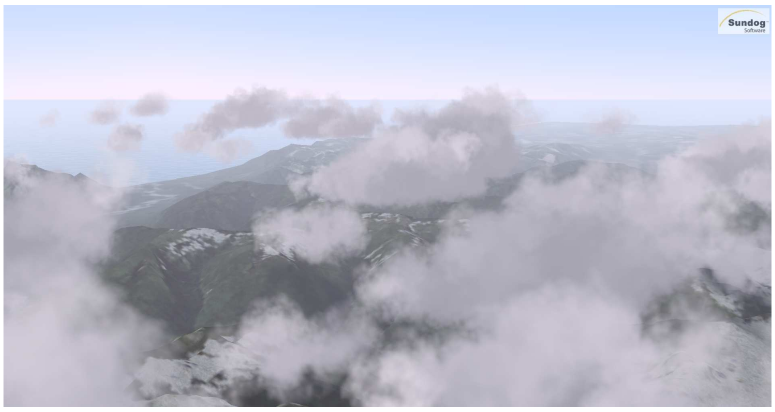
In general, particle systems can be used for either physically-based or procedural approaches, although within the scope of cloud simulation there appears to be little physically-based work besides that which is presented here [McLoughlin 2008]. Perhaps the closest was the particle-type "qualitative simulation" used by Neyret [1997], which used elements of physically-based techniques. In this work, parcels were simulated as “bubbles” that could rise in turrets, and which together formed the surface of the cloud, using processes designed to mimic the visual appearance and growth of newly formed cumuliform clouds. Initial results, although simply rendered, were promising but unfortunately the work remains incomplete.

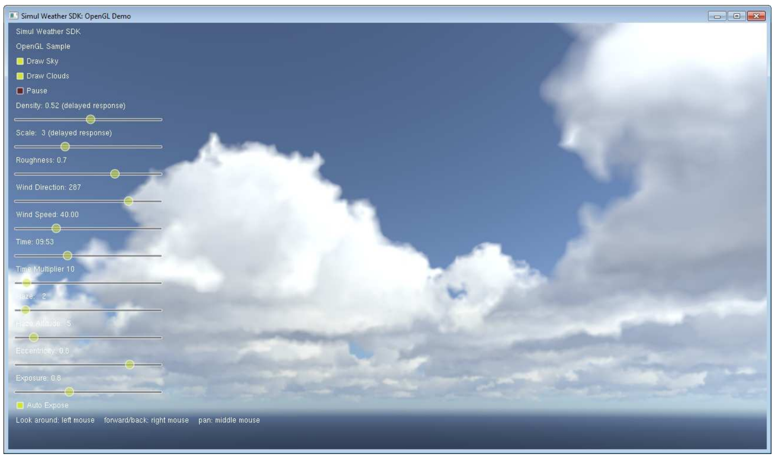
Particles were successfully used to generate clouds by Harris and Lastra [2001], who dynamically rendered groups of particles to texture, and applied this texture to *impostor* sprites. Unfortunately, they did not provided full details on how the particles’ initial positions were determined.

An image-based approach has also been taken by Dobashi et al. [1998b], who used satellite imagery to create a metaball description of real cloud systems.

3.2.3.1 Middleware Solutions

There currently exist at least two third-party products that developers can use for the addition of clouds and weather effects into a game. The first of these is Silverlining, by Sundog Software [Sundog Software ca.2008], which uses a cellular-automata method for simulating cloud growth, and includes a mixture of procedural techniques for cloud generation with physically-based aspects. It appears to represent clouds with a number of small sprites, and its interface offers control over cloud generation by giving the developer the ability to specify ‘cloud layers’ of specific cloud types. The cloud types supported are: two forms of cumulus, cumulonimbus, stratus and cirrus, and the system also supports precipitation in the form of rain, sleet or snow. Individual clouds are then generated within the specified cloud layer. The system also includes a cloud-lighting model, as well as support for lightning. The second product is Simul Weather, by Simul Software [Simul Software ca.2008b]. This simulates a cloud layer using what appears to be multiple octaves of 3D procedural noise, although the authors claim it uses a physically-based technique for cloud generation. Visual results from both middleware solutions are shown in Figure 3.1. Currently, neither of these product solutions are likely to be a large improvement over the vast majority of cloud techniques used in modern computer games, in this author’s opinion. Alternatively, the developers of Simul Weather also offer a product, CloudWright [Simul Software ca.2008a], which can be used to produce a static image for use with a traditional sky box or sky dome.




Simul Weather SDK: OpenGL Demo

Simul Weather SDK

OpenGL Sample

🟨 Draw Sky

🟨 Draw Clouds

🟥 Pause

Density: 0.52 (delayed response)

Scale: 3 (delayed response)

Roughness: 0.7

Wind Direction: 287

Wind Speed: 40.00

Time: 09:53

Time Multiplier 10

Haze: 2

Haze Altitude: 5

Eccentricity: 0.8

Exposure: 0.8

🟨 Auto Expose

Look around: left mouse   forward/back: right mouse   pan: middle mouse

Figure 3.1: Above, screenshot from the interactive demo of Silverlining by Sundog Software [Sundog Software ca.2008]. Below, screenshot from the interactive Simul Weather SDK demo by Simul Software [Simul Software ca.2008b].

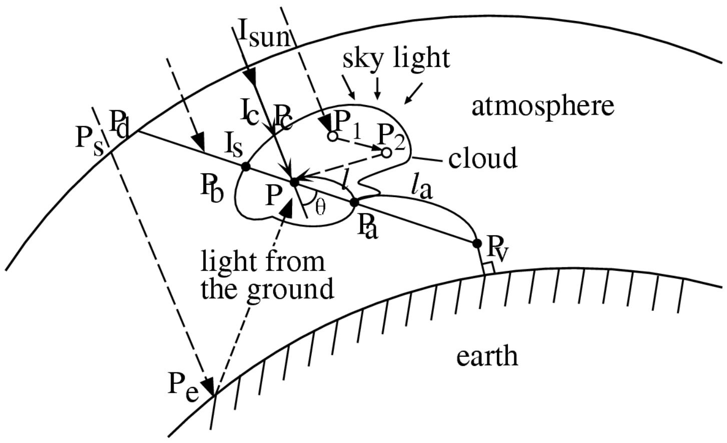


Figure 3.2: The paths that light can take, and which should be evaluated, when rendering clouds. Diagram taken from Nishita et al. (Figure 1) [1996]

3.2.4 Rendering-Specific Work

A great deal of work has been undertaken in the subject of cloud rendering. However, the emphasis of this thesis is on cloud simulation, so what follows is only a very brief glance over the topic:

The path that light takes through participating media is complex. Figure 3.2 shows the incident light on a single particle (P) within a cloud, and includes direct illumination from the sun (I_{sun}), light from the sun indirectly received due to the scattering of other particles (path P_1P_2P), illumination from the sky (itself due to scattering of sunlight by particles within the atmosphere) either directly if the particle is at P_b or indirectly due to scattering by other cloud particles, light reflected from the ground, and finally attenuation due to other atmospheric particles (along the length l_a), all of which must then be determined for "every particle along the viewing ray" [Nishita et al. 1996]. This is clearly not a trivial problem.

Approaches to solving this have included the simulation of single scattering events [Blinn 1982; Dobashi et al. 2000] and multiple scattering events [Kajiya and Herzen 1984; Nishita et al. 1996; Harris 2002]. Blinn [1982] also described the use of a phase function, which defines how illumination can change with respect to the angle between the light and the viewing vector (see Figure 3.3). Precomputation methods have also been tried although, for example in the case of the work by Zafar et al. [2006] who used spherical harmonics, the precomputation times of 47 minutes hardly make the technique suitable for real-time dynamic simulations.

Various rendering techniques have also been applied to meteorological data-

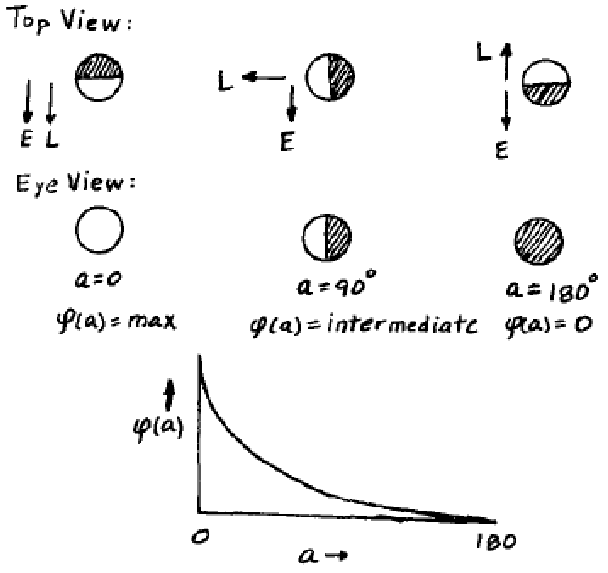


Figure 3.3: The phase function defines the illumination characteristics of a particle as the angle between the light and the viewing vectors change. Image from Blinn (Fig 3) [1982].

sets, beyond those directly employed in meteorology [Manssour et al. 1996; Vis5D ca.2012]. Papathomas et al. [1988] provided summaries of an extensive range of early works. A traditional approach for scientific visualisation is to create some form of surface from the volumetric data that is produced, and then to render this [Trembilski and Brossler 2002a; Trembilski and Brossler 2002b]. The goals of such approaches are usually to identify specific aspects of the data, and can also include the flow itself [Cuntz et al. 2007]. Alternative works have tried to produce a visually realistic representation of the data. Riley et al. [2003; 2004a; 2004b] used different transfer functions for the various different water states, including one for each colour component, and could even simulate coronas, haloes, rainbows and glories. Many of these optical effects have also been addressed in the purely graphics related work by Bouthors et al. [2006], who made use of the GPU and even included a radiosity algorithm to account for inter-reflections between the cloud layer and the ground surface.

3.2.5 Other Weather-Related Effects

Traditional methods of rendering rain involve either large numbers of particles, or scrolling-textures applied to sprites [Wang and Wade 2005]. The problem of rendering individual raindrops has been studied in detail by Garg and Nayar

[2006], and their technique has also been ported to a GPU-based particle system [Tariq 2007], although the implementation was only a subset of the full solution. Additionally, Wang and Wade [2005] used textures applied to ellipsoids, while Tatarchuk and Isidoro [2006] used an image-based solution that made heavy use of the GPU, and Wang et al. [2006] incorporated a pre-computed radiance transfer method for improved lighting in a particle-based approach. Methods have also been developed for the effects of rain on surfaces [Tatarchuk and Isidoro 2006; Stuppacher and Supan 2007], as well as the accumulation of snow on objects [Fearing 2000]. Several attempts have also been made at lightning effects [Dobashi et al. 2001; Kim and Lin 2004].

Volumetric attenuation, due to haze or fog, has also been a popular topic of research and of interest for games [Sun et al. 2005; Wenzel 2006], and Dobashi et al. [2000] visualised the shadow of clouds in an attenuated atmosphere, while the colour of the sky itself has also been examined [Preetham et al. 1999].

3.2.6 General Fluid-Dynamics Work

An introduction to the topic of fluid dynamics can be found in Appendix A, which covers the fundamental terms and techniques as well as the primary works within computer graphics.

Early work in fluid dynamics within computer graphics was mostly hampered by the computational power required to solve the Navier-Stokes equations — for example, for the film *2010*, Yaeger et al. [1986] had to use a Cray X-MP supercomputer to simulate a 2D flow that advected textures of Jupiter's cloud system. Since then, computing power has thankfully increased to the point where today simple fluid-dynamics can be performed at interactive rates. As mentioned in Section 2.7 and Appendix A, the subject of fluid-dynamics is extensive and the topic of much active research. Only a small proportion of this work, however, is directed toward the problem of simulating the atmosphere, and cloud systems within it. Water simulations often involve particular forms of the motion equations, and tend to deal with height-maps that are of limited relevance to atmospheric approaches [Kass and Miller 1990]. Smoke simulations, on the other hand, are highly relevant, even if they are often treated more as passive travellers in a flow — there being no water to change between states, and hence no additional heating effects due to state changes:

Foster and Metaxas [1997] presented to the graphics community a three-dimensional Eulerian approach that used the Navier-Stokes equations and employed a solution from previous computational fluid-dynamics literature [Foster and Metaxas 1997 cite Harlow and Welch 1965] . However, early solutions were susceptible to numeric instabilities and perhaps the most significant contribution to the subject was by Stam [1999], who developed a stable form of the equations, as discussed in Appendix A. In general, most Eulerian-based techniques within computer graphics now use this method as a basis, although computational costs are still high. The work by Kim et al. [2008] used procedural techniques to add high-frequency turbulence to a low-resolution fluid solution, using a wavelet decomposition scheme to determine where detail should be added to the flow.

Fluid-dynamics techniques have been implemented on graphics hardware [Harris 2004; Goodnight 2007], primarily due to the fact that the hardware is specifically designed for the parallel execution of operations on grid-based structures (i.e. textures and frame-buffers)[3], which can be used as Eulerian grids.

Lagrangian techniques have also been developed [Miller and Pearce 1989; Premoze et al. 2003]. Although early particle works on the GPU were limited to stateless systems [Fernando and Kilgard 2003, Section 6.3, p. 149], full particle systems were quickly developed [Latta 2004; Kolb et al. 2004], and Lagrangian fluid dynamics techniques are now possible [Harada et al. 2007].

A popular Lagrangian approach is *Smoothed Particle Hydrodynamics* (SPH), a method originally developed by astrophysicists [Lucy 1977; Gingold and Monaghan 1977]. In this technique, particles exert forces upon each other according to a kernel function which varies with distance. Desbrun and Cani [1996] are credited (by Bridson [2008, p. 139]) as introducing this technique to the graphics community where it has since been used for special effects [Horvath and Illes 2007], and has been extended [Müller et al. 2003; Losasso et al. 2008], and implemented on the GPU [Kolb and Cuntz 2005; Harada et al. 2007].

A number of works have also developed Lagrangian vortex methods in 3D, which involve some form of approximation for the vortex stretching term (see Appendix A, Section A.3.2 on page 148) [Selle et al. 2005; Park and Kim 2005; Angelidis et al. 2006].

Hybrid methods also exist, which use both Lagrangian and Eulerian elements. One such example is the 'Particle-In-Cell' approach [Bridson 2008, p. 147]. Alternatively, a standard Eulerian fluid simulation method can be used to drive a particle system [Oleg A. Potiy 2005].

The problem of artistic control has also been addressed for fluid-dynamics in general [Treuille et al. 2003; McNamara et al. 2004; Kruger and Westermann 2005; Kim et al. 2006; Zhou et al. 2006]. Such techniques typically work through the application of additional body forces that carefully adjust the flow into a desired shape.

[3]While graphics primitives are specified in terms of vertices and polygons, the number of vertices is generally small relative to the number of pixels that are rasterised and operated on, so graphics hardware has traditionally accelerated per-pixel operations to a greater extent. The exception to this, of course, is that GPUs are becoming more generalised, and programmable units can now be assigned to either per-vertex or per-pixel operations.

Chapter 4

Cloud and Weather Simulation

This chapter presents the main contribution of this work. Here we will look at how the solutions were developed, their details and results. We will start by looking at a number of initial approaches, which were valuable experiments for informing the design of the final solution.

4.1 Initial Experiments

A simple cloud technique was first investigated, for use in a game-like environment [Anderson and McLoughlin 2007]. This served as an experimentation platform and inspired the main body of work, details may be found in Appendix B. Following this, a parcel model was developed based on the techniques reviewed in the Chapter 2 of this thesis.

OpenGL has been used for all of the prototype systems presented here, and discussions will use OpenGL terminology (where appropriate) in preference over other API terms.

4.1.1 A Parcel Model

A parcel model was developed and implemented, following the theories set out in Chapter 2 of this thesis [McLoughlin 2008].

4.1.1.1 2D Vortex-based Fluid Dynamics

In order to test the parcel model, a fluid-dynamics system was required to simulate the motions of the atmosphere. Rather than directly solving the Navier-Stokes equations, an approximation for 'fluid-like' behaviour was pursued. The solution was a crude two-dimensional vortex particle approach which demonstrated the desired behaviour.

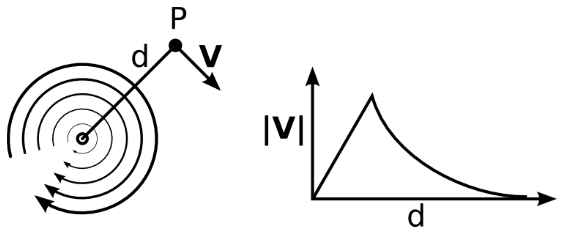


Figure 4.1: Velocity is determined at a sample point P, where the magnitude is a function of distance. A Rankine vortex function is shown here.

4.1.1.2 Vortex Particles

The vortex particles developed here are not the same as the vortices described in Section A.3, which were microscopic rotations in a vector field. Instead, the vortices used here are macroscopic 'point vortices'. They are defined by a vortex function, which is directly sampled to obtain a velocity vector. This function is also modulated by a general strength parameter and scaled to an effective radius. There is thus no need to evaluate the stream function, and solve the associated equations. (See Figure 4.1).

4.1.1.3 Vortex Particle Emission

One of the major issues with any Lagrangian vortex technique is the initial placement of the vortices. Here, the vortices will ultimately be assumed to be generated by the movement of a parcel through the fluid. As an approximation to the non-slip condition, vortices are generated in pairs on either side of the parcel when it moves. In the case of initial prototypes, one parcel was present and assumed to be under the direct control of the user via the mouse cursor.

4.1.1.4 Particle Interaction

Interaction with the point vortices is achieved by using the sample location to evaluate the vortex function from all vortices active within the simulation domain. The result of this is a velocity vector, which can be used to advect either another vortex particle, parcels or tracer particles placed in the flow. The advection scheme used was forward-Euler, which naturally resulted in a potentially unstable simulation, so careful adjustment of the vortex strength and radius were required to obtain stable results.

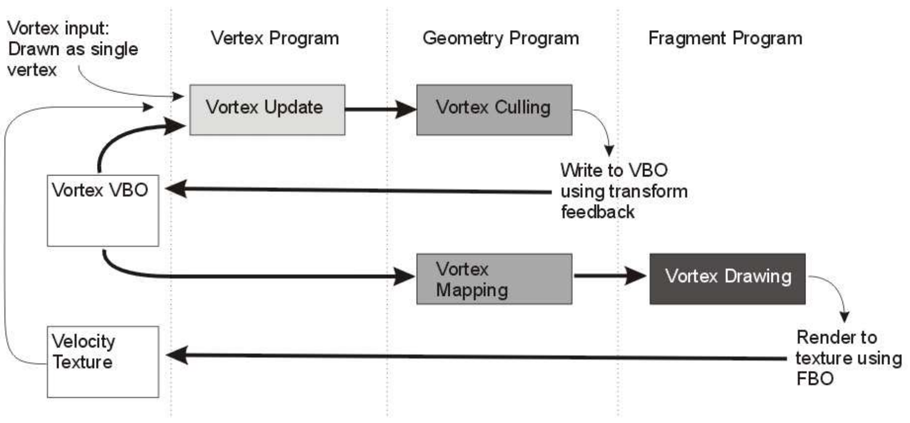


Figure 4.2: Operations in the 2D vortex technique

4.1.1.5 Implementation

The system was initially implemented on the CPU, but poor performance warranted its conversion to the GPU where it was far more effective. This was achieved using Cg [Fernando and Kilgard 2003] and general-purpose GPU techniques. The basic steps of the GPU implementation are as follows:

- Apply a vortex-update operation to all new and (old) stored vortices, updating:
  - Positions, by sampling a velocity texture;
  - Age;
  - Strength;
- Cull any vortices that either exceed a lifetime threshold, or are below a strength threshold;
- Store all remaining vortices, for use in the next iteration;
- Write the vortices' functions to a velocity texture, for updating positions in the next iteration.

The final step is performed by drawing the vortices as sprites with the vortex velocity function pre-calculated as a texture. The vortices themselves are stored in vertex buffer objects (VBOs) so this is achieved either through the point-sprite mechanism, or a simple geometry shader that generates a primitive for the texture. The previous steps can make use of the transform feedback extension, allowing arbitrary numbers of vortices to be created. If the fragment shader were

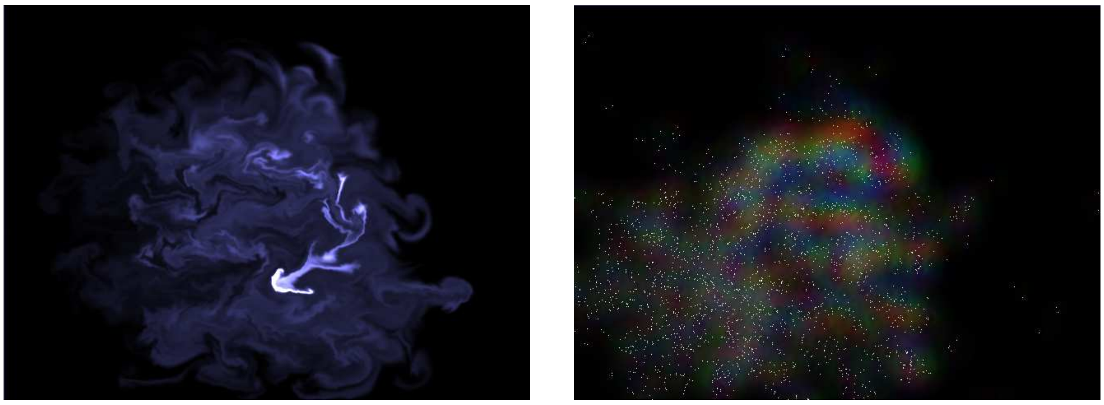


Figure 4.3: The 2D vortex technique in action, within a simple application. Left, the user interacts with the ‘fluid’ by using the mouse to inject ink. Right, the vortex particles and a visualisation of the velocity texture.

to be used, the number of vortices would be fixed to those that would fit into a texture. While it is then possible to add a parameter to indicate the activation state, an inactive vortex would still be processed by the shader, leading not only to a minor increase in memory usage but also to redundant computational exertion. The operations are shown in Figure 4.2.

4.1.1.6 Results

Typical visual results from a test application are shown in Figure 4.3. This application used an additional framebuffer-object bound texture, in which an ‘ink’ colour was added according to the position of the mouse. The velocity texture, output from the vortex technique, was then used to advect the ink. Rankine vortices [Acheson 1990, p. 15] were used in this example.

Frame-rates achieved were typically in excess of 38 fps at a resolution of 640 by 480, on a dual 3GHz PC with an NVIDIA GeForce 9800GX2 (not in SLI mode). Because the fluid technique is largely resolution independent, the test application scaled well to higher resolutions, achieving 33 to 35 fps at a resolution of 1920 by 1200. The maximum number of vortices in both cases was 16384. Because the technique was designed to only visually approximate the physical processes of fluid dynamics, it was considered unnecessary to perform a detailed quantitative analysis to determine the accuracy of the solution.

4.1.1.7 Bulk Water Continuity

For this initial work, the parcel theory that was outlined in Chapter 2 of this thesis was used with simplified cloud microphysics. The result was a lightweight bulk-water continuity model:

Three major water categories are defined as cloud vapour q_v, cloud water q_c, and precipitation ready water q_p. To reduce the microphysical considerations, only two source / sink factors are defined: the vapour to cloud source S_{vc} and the cloud to precipitation source S_{cp}. Thus:

$$\{q_v\} \underrightarrow{S_{vc}} \{q_c\} \underrightarrow{S_{cp}} \{q_p\}$$

Where the direction of the arrow indicates the positive direction, but flow naturally can occur in both directions.

The cloud and precipitation water categories are then subdivided into the liquid and solid states, cloud liquid q_{cl}, cloud ice q_{ci}, precipitation rain q_{pr}, precipitation snow q_{ps}, and precipitation hail q_{ph}. Thus:

$$\{q_v\} \underrightarrow{S_{vc}} \left\{ \begin{array}{c} q_{cl} \\ q_{ci} \end{array} \right\} \underrightarrow{S_{cp}} \left\{ \begin{array}{cc} q_{pr} & \\ q_{ps} & q_{ph} \end{array} \right\}$$

The equations for S_{vc} and S_{cp} are:

$$S_{vc} = \max\left(-q_c,\, q_v - q_{vs}\right)$$

$$S_{cp} = \max\left(-q_p,\, q_c - q_{ct}\right)$$

where q_{ct} is a ‘cloud threshold’ value, above which cloud water turns to precipitation ready water and q_{vs} is the saturation vapour mixing ratio, as defined by Equation 2.4.2.1 on page 41.

The sub-states of cloud water and precipitation ready water will transfer their water values between each other, according to the current temperature of the parcel. The above formulations imply that water transformation between states occurs instantaneously, which is certainly not the case physically. An additional factor was therefore applied to allow the transformations to take place over a small amount of time (experimentation showed that typically one second was sufficient).

4.1.1.8 Implementation

The background atmospheric description was implemented as a one-dimensional texture, which stored the temperature and pressure gradients. The motivation behind this was to provide a level of control over the system, since the temperature could then easily be modified to any desired profile.

In addition to the vortex-particle motion scheme previously described, the steps involved in the parcel simulation method were:

- A parcel-update operation, applied to all new and (old) stored parcels, by updating:
  - Water states, according to the continuity equations above — however, only the warm equation set was implemented for this system;
  - Temperature, as a result of water state changes (see Equation 2.4 on page 44);
  - Buoyancy, based on the difference between parcel temperature and the background atmosphere (see Equation 2.5 on page 47);
  - Positions, by sampling the velocity texture generated as output from the dynamics simulation subsystem, added to the buoyancy;
  - Age;
- Culling any parcels that exceed a maximum age;
- The storage of all remaining parcels.

The operations for the combined fluid solution and cloud simulation system are shown in Figure 4.4.

Parcel generation was handled by a seeding mechanism which could generate a number of parcels in close proximity at the ground level, to emulate ground heating effects. Different categories of these were designed to initialise parcels with different properties, resulting in the formation of different cumulus sizes. Further, one type generated a series of very warm and moist parcels from one side of the simulation domain to the other, in an attempt at generating a cumulonimbus-like formation. These parcel seeding mechanisms were generated randomly during simulation time.

Parcel visualisation was achieved using a two-dimensional metaball style technique, in which each parcel added a kernel function (provided by a pre-generated texture) to a framebuffer-object bound texture. This was then visualised.

4.1.1.9 Results

The visual results of the system are shown in Figure 4.5. The prototype successfully demonstrated the formation and evolution of simple two-dimensional ‘cumuliform clouds’, including their ability to produce rain.

Typical performance of the system with 6000 parcels and 25000 vortices was 25 fps, and for 15000 parcels and 50000 vortices was 13 fps on an NVIDIA GeForce 9800GX2. The fluid-like solution was adequate for testing the parcel system, but was not ideal and required a high level of fine-tuning to achieve satisfactory results, as well as a high proportion of vortices in relation to the number of parcels.

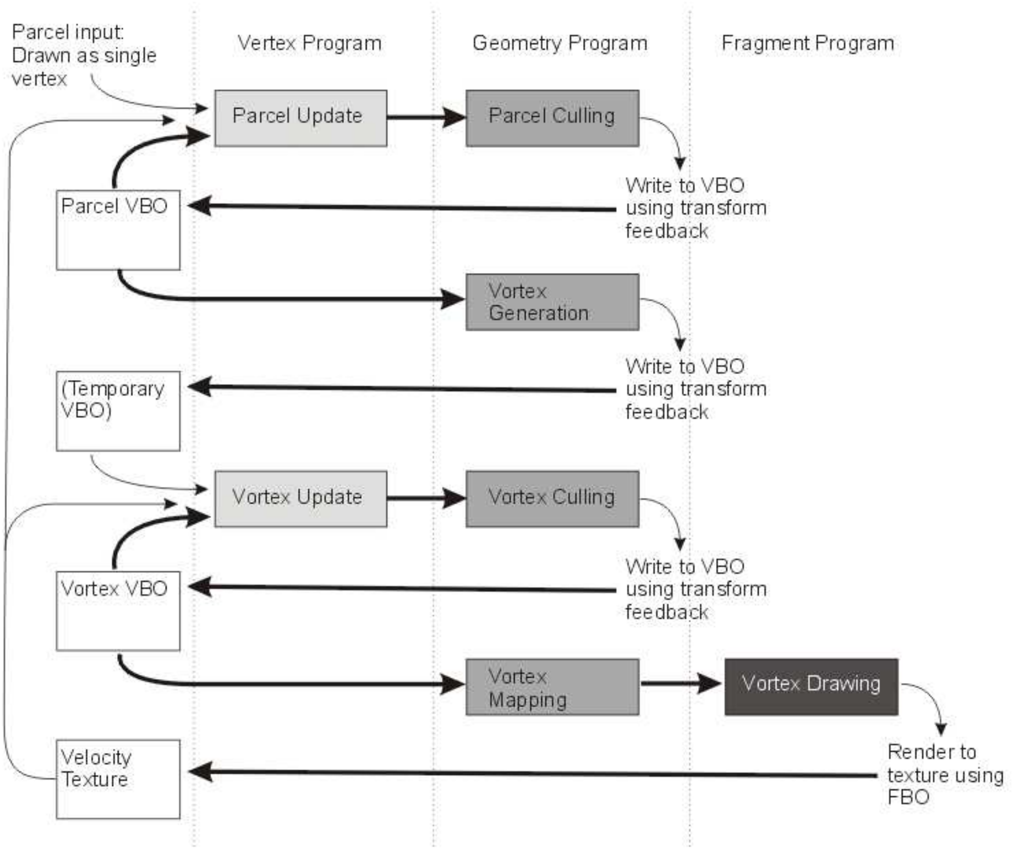


Figure 4.4: Operations diagram for the combined system.

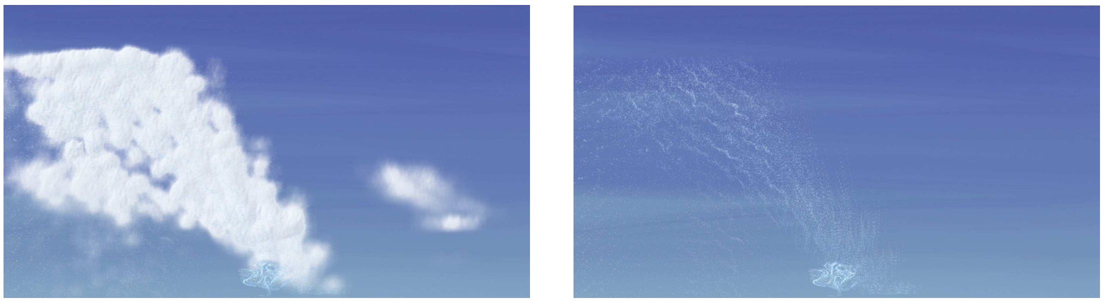


Figure 4.5: Visualisation of the 2D cloud-parcel system, with 21732 parcels and 50000 vortices. Left, parcels and vortices. Right, vortices only.

4.2 Main Simulation

The basic processes of cloud formation have been described in Chapter 2 and a graphical overview of the cumuliform cloud formation process is shown in Figure 4.6.

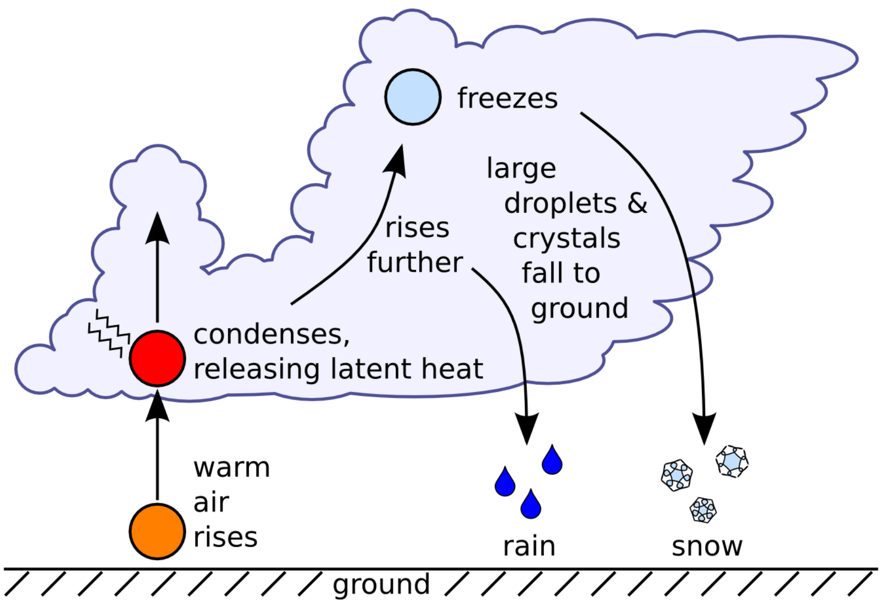

freezes
large droplets & crystals fall to ground
rises further
condenses, releasing latent heat
warm air rises
rain
snow
ground

Figure 4.6: Overview of the cumuliform cloud formation process.

For the main simulation the Eulerian approach was taken, where the simulation domain is described by a three-dimensional lattice or grid (hereafter grid) and each grid cell contains a number of properties. The equations of fluid dynamics describe how these properties move from one cell to another, while the various cloud and water equations describe how the properties within each cell change and interact with each other. The equations that make up the water model are essentially one-dimensional, being spatially dependent only on height. The technique chosen to model the fluid properties of the atmosphere can therefore be considered as an independent problem. As demonstrated with the initial experiments, such a model is suitable for implementation using an Eulerian solution, a Lagrangian one or some hybrid approach. An overview of the design is shown in Figure 4.7.

In this section, the work from Chapter 2 will be brought together to build the final cloud model. The key points and equations will be reiterated for completeness and aspects of the model that are different to existing works within computer graphics will be highlighted. To help define the cloud equations we shall continue with the convention of the parcel model. As described in Section 2.3.2 on page 34,

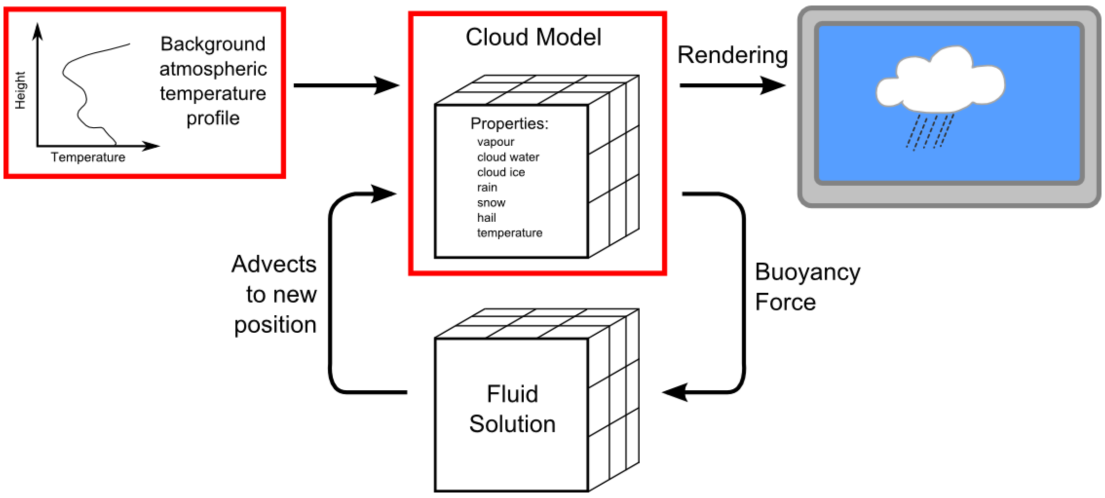


Figure 4.7: The cloud model is conceptually separated from the fluid solution. The two aspects this work focuses on are highlighted in red.

parcels are discrete traceable volumes of air which are free to travel through the atmosphere but do not mix with it. The boundary of the parcel is considered not to conduct heat and, provided the parcel itself does not generate or absorb heat, any processes performed on or by the parcel, such as movement, are termed *adiabatic*.

4.2.1 Background Atmosphere

We assume that parcels move through a static 'background' atmosphere which is defined by temperature and pressure, with density dropped from the equations. As a rough generalisation, temperature decreases with height within the troposphere (the first approximately 10km layer of the atmosphere where clouds form). In reality the rate of decrease actually varies quite widely, which can influence cloud formation. In contrast with previous works that use a fixed rate, this thesis introduces the use of a user-defined temperature gradient. The advantage of this variable temperature gradient is that it will give us greater control over the cloud formation, as we shall see shortly in Section 4.2.3.1 on page 82.

The pressure of the atmosphere also decreases with height and can be modelled using hydrostatics. The following approximation is used to calculate the pressure P at height h: [Bohren and Albrecht 1998, p. 55]

$$P \approx P_0 \exp\left\{-\frac{h}{H}\right\}$$

where P_0 is the base pressure and H is the *scale height* constant, set to 7.29 in this study as defined by Bohren and Albrecht. This equation has the advantage over the more traditional barometric formula [Rogers and Yau 1989, p. 28] in that it is independent of temperature, which simply makes calculations easier when using an arbitrary temperature profile by removing the need to integrate the temperature up to the sample height.

4.2.2 Parcel Properties

The pressure of a parcel is considered to equalise rapidly to its surroundings and so follows the background atmospheric pressure profile. As a parcel moves through the atmosphere adiabatically its pressure will therefore change and, from the ideal gas law, it should be clear that a change in pressure must result in a change in either density, temperature, or both. Since we are ignoring density, we assume that this change occurs to the parcel's temperature. The temperature property of the parcel is stored as its potential temperature, which is constant during adiabatic processes: [Houze 1993]

$$\theta \equiv \left(\frac{\hat{p}}{p}\right)^{\kappa} T$$

where κ is a function of gas constants approximately equal to 0.286, p is the pressure of the parcel, $\hat{p}$ the reference pressure and T is its actual temperature.

The parcel description also includes a set of water properties, which are stored as *mixing ratios* — the mass of water per mass of air and, although dimensionless, they are usually expressed with their units of either kg/kg or g/kg. The water present in a parcel is then further subdivided by category, according to those defined by the water model which is used. This thesis proposes a six-state model, so a parcel stores mixing ratios for: vapour q_v, cloud liquid q_{cl}, cloud ice q_{ci}, rain q_r, snow q_s, and precipitating ice (hail/sleet/graupel, hereafter just hail) q_h. Here 'cloud' liquid and ice are considered to consist of particles small enough that they remain suspended in the air, whereas rain, snow and hail particles are sufficiently large that they fall out of the cloud. The exact processes involved in the transfer of water between these categories will be described in Section 4.2.4. For now though, it is useful to note that water vapour condenses into liquid when the parcel becomes 'saturated' and that the mixing ratio at which saturation occurs is a function of pressure (which we have just defined as a function of height) and temperature.

The key aspect which links the parcels and the background atmosphere to the fluid motion system is buoyancy, the movement which results when gravity acts upon differences in density between a volume and its surroundings. Again following the trend of removing density from the equations, it can be shown [Emanuel 1994, pp. 6-8] that the resulting force due to buoyancy for a parcel of unit mass is related to the difference in temperatures between it and its surroundings (i.e. the background atmosphere). Further, gravity will act upon any water that is present although in the case of small particles this is not large enough to make them fall from the cloud. These factors may be combined into the following equation for the force due to buoyancy: [Rogers and Yau 1989, p. 50]

$$F_B = \mathbf{g}B = \mathbf{g}\left(\frac{T}{T'} - (1+q)\right)$$

where $\mathbf{g}$ is the gravitational acceleration vector, T is the parcel temperature, T' is the ambient environment temperature taken from the background atmospheric description and q is the parcel's total mixing ratio of all condensates (all water categories except vapour).

4.2.2.1 Wind

Wind forces play a vitally important role in shaping a cloud formation. The detailed examination of these effects are beyond the scope of this work, although a simple scheme is provided. The first step is the provision for high-frequency, low-amplitude turbulence which is implemented as a pseudo-random vector field that is added as an additional body force in the advection step of the fluid-dynamics system. The treatment of wind forces loosely assumes that they are caused by larger-scale effects than the simulation domain physically covers. A vector field is defined by the user by applying simple primitives, such as a uniform wind that applies a specific vector within its bounds. This vector field is again used to apply a body force to the fluid.

4.2.3 Temperature Profile

Previous works within computer graphics have all used a fixed-rate for the background atmospheric temperature profile. We shall now examine how the inclusion of an arbitrary temperature profile can lead to more visually interesting and controllable results and present a graphical tool for setting the temperature profile.

As described in Section 2.5 on page 47, regions of atmosphere may enhance, inhibit or be indifferent to a parcel's vertical ascent. To summarise: a rising parcel's temperature will initially decrease according to the dry adiabatic lapse rate (DALR). When condensation occurs, the latent heat causes the parcel to warm slightly so its temperature decreases according to the saturated adiabatic lapse rate (also called the pseudoadiabatic or moist lapse rate). If the background atmosphere's lapse rate is greater than the DALR, a rising dry parcel's buoyancy will increase and the atmosphere is said to be unstable. If the atmospheric lapse rate is identical to the DALR a dry parcel's buoyancy is zero and the atmosphere is said to be neutral. If the atmospheric lapse rate is less than the DALR a rising dry parcel will exhibit a gradual decrease in buoyancy until it reaches an equilibrium altitude and the atmosphere is said to be stable. If the atmospheric lapse rate is less than the DALR but greater than the pseudoadiabatic lapse rate, a dry parcel experiences a region of stability while a moist parcel experiences a region of instability, so the region is termed conditionally unstable.

In this thesis, it is proposed that we can direct cloud formations by manipulating the background atmospheric temperature profile to include regions of stability and instability, thus influencing the buoyancy of parcels and the resulting cloud formation.

4.2.3.1 Reverse Skew-T

As discussed in 2.5.1 on page 48, the skew-T is primarily an analysis tool for determining properties of a given temperature profile, usually measured by radiosonde (measuring devices attached to weather balloons which relay information to a ground station via radio as the balloon rises). However, in this work we need to perform the reverse of this, by starting with desired characteristics of the atmospheric profile and using them to generate the profile itself.

The premise of the technique is that it would be useful to be able to specify a cloud base and a cloud height and from these to determine the atmospheric temperature profile and the required temperature and moisture of a source that will generate the cloud. Using the skew-T this is a relatively straight-forward problem. Perhaps just as important, if the temperature profile were also presented on the chart, a more advanced user could adjust it to give finer control.

The requirement then is for the tool to create an atmospheric profile with a region below the cloud base which is unstable for a warm parcel and will cause it to rise. In the region from the cloud base to the cloud top, the parcel needs to continue its ascent, so this region should also be unstable. Finally, the region above the cloud top needs to be stable to prevent the parcel from rising above it.

The definition of the cloud base is a two-dimensional problem, which not only requires a height but also a temperature. In the prototype tool, rather than assuming a default temperature this option is specified by the user who may have a preference, for example for a cold cloud mostly made of ice over a warm one made of liquid water. Looking at the skew-T graph in Figure 4.8, the cloud base can easily be specified by a single point, which is the LCL (lifted condensation level). From this point we can read off the potential temperature line to find the potential temperature that a parcel must have in order for it to reach the desired temperature at the specified height. Similarly, if we read off a mixing ratio line, we can find the amount of water it needs to possess such that when it reaches this height it will become saturated and start forming a cloud.

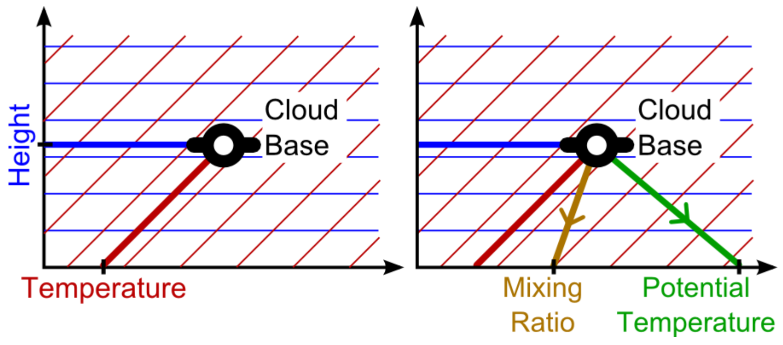


Figure 4.8: Left, by specifying the height and temperature of the base of the cloud, we can use the skew-T to determine the parcel's required potential temperature and water mixing ratio as shown on the right. Please refer to Figure 2.11 on page 49 for the base skew-T axes.

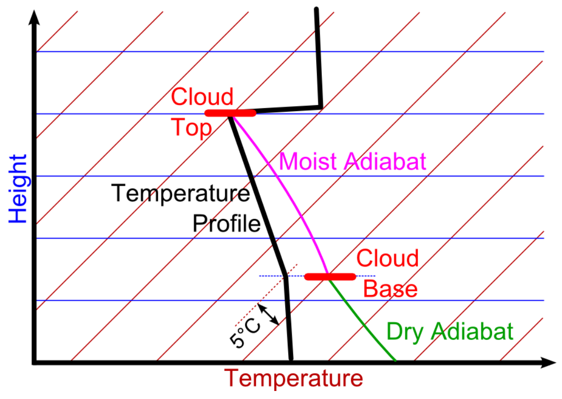


Figure 4.9: The 'reverse' skew-T solution for generating a background atmospheric temperature profile that ensures cloud formation between a user-specified cloud base and top. A fixed lapse-rate is used up to the altitude of the cloud base, but at a temperature 5 degrees Celsius lower. Above this, the profile aims to meet the cloud top altitude and temperature. Above the cloud top is a region of high stability.

Although a wide variety of approaches are suitable for solving the problem, the complete solution used in the prototype tool was to employ a simple set of lapse-rates (see Figure 4.9). For the region below the cloud base, a rate of $7\,K.km^{-1}$ was used, which is below the dry adiabatic lapse rate and is stable for a rising parcel of the base atmospheric temperature. However, this lapse rate was offset by starting five degrees colder than the cloud base, so a parcel of the specified potential temperature will not reach its equilibrium point and will rise through this region.

The region between the cloud base and top, in which it is intended that clouds will form, uses a lapse rate which ensures that the profile meets the parcel's pseudoadiabatic curve at the cloud top height. This lapse rate is itself less than the pseudoadiabatic lapse rate, and so is stable for a parcel starting at the cloud base height but with the profile's temperature. The advantage of this is that if we assume a number of parcels will actually be produced, ranging in potential temperature *up to* the specified value, they will stop below the maximum height and result in a deep cloud that lies between the specified base and height.

The region above the cloud top uses a lapse rate of $5\,K.km^{-1}$, following from the cloud base temperature. This is well below the dry and pseudoadiabatic lapse rates and so provides a highly stable environment in which the cloud will experience negative buoyancy. This sharp region of high stability ensures that parcels quickly stop their ascent.

4.2.4 Six-State Water Model

This section presents details of a six-state *bulk water continuity* model. This describes how water is transferred between states whilst conserving total water mass of non-precipitating states, and methods for dealing with water transfers between precipitation states.

Most water continuity models, such as that by Lin et al. [1983], include detailed approximations to complex physical processes whereas the method here is lightweight and tailored specifically to the requirements of computer graphics. It is an adaptation and extension of the work by Schultz [1995] with the introduction of a set of control parameters for manually adjusting some of the transfers and with further simplifications of the equations. In his model, Schultz uses *specific contents* as the measure of water for many equations, which are a mass of water per volume and will change when the parcel's density changes. Although less physically accurate, the work presented here solely uses mixing ratios because these are conserved during adiabatic processes.

Although Schultz's methods include many control parameters, they are intended to allow the model to be adjusted to fit observed data and thereby improve accuracy. The intention of this work is not to produce an accurate simulation per-se, but to produce a visually believable one which offers a level of control to the user. To this end the parameters in this work are intended to allow the user to adjust the simulation to provide an output that suits their own goals. Further, the parameter values may also be varied spatially, for instance using a texture map, rather than acting as global values.

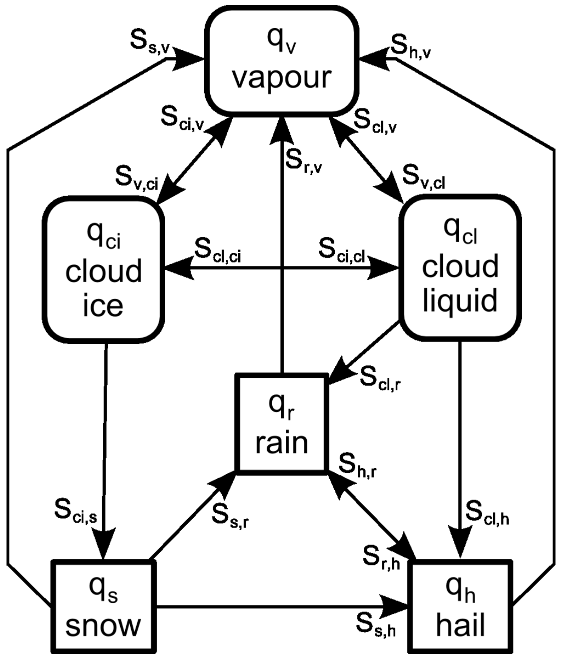


Figure 4.10: The six-state bulk water continuity model showing the categories and transfers.

The complete set of state transfers are shown in Figure 4.10. Each transfer is described using a source / sink parameter S, with subscripts denoting the source and destination categories. For example, the vapour to cloud liquid transfer parameter is $S_{v,cl}$. The change to each category's water mixing ratio in a given time step is then the sum of all sources and sinks:

$$\begin{aligned}
\Delta q_v &= S_{cl,v} + S_{ci,v} + S_{r,v} + S_{s,v} + S_{h,v} - S_{v,cl} - S_{v,ci} \\
\Delta q_{cl} &= S_{ci,cl} + S_{v,cl} - S_{cl,ci} - S_{cl,r} - S_{cl,h} - S_{cl,v} \\
\Delta q_{ci} &= S_{cl,ci} + S_{v,ci} - S_{ci,cl} - S_{ci,s} - S_{ci,v} \\
\Delta q_r &= S_{cl,r} + S_{s,r} + S_{h,r} - S_{r,h} - S_{r,v} - G \\
\Delta q_s &= S_{ci,s} - S_{s,r} - S_{s,h} - S_{s,v} - G \\
\Delta q_h &= S_{cl,h} + S_{r,h} + S_{s,h} - S_{h,r} - S_{h,v} - G
\end{aligned}$$

where all parameters are clamped to positive values and G is a ground factor which simply removes all precipitation that reaches the ground. The primary purpose of this section is to define equations for each of these parameters — this is not as overwhelming a task as it may initially seem because each parameter appears twice and some are very similar to each other. Care must be taken with large time steps to maintain total water mass. The maximum value that each parameter can take is simply the current quantity of water present from the source of the transfer. Thus, because parameters are capable of removing the entire water content from a category they must be evaluated separately and in some cases in a specific order. Following Schultz, a number of the transfers are assumed to be instantaneous, while others occur over time. If the time-step between frames is small, the instantaneous processes can also be easily modified to occur slowly over time. Likewise, if the timestep is large, some of the timed-conversions will occur instantly. While the effects of some conversion rates and other constants are subtle, others have more obvious effects which will be identified as they are presented. Except where specified, the order in which the parameters are evaluated is not particularly significant.

All of the parameters listed above contain control variables, which are each designed to constrict flow *into* a specific state to give the user control over the water transfers. There are thus six control variables which each occur in a number of different source / sink parameters. The control variables are denoted C with a subscript identifying the state they control flow into. Thus C_{ci} controls the flow from any category into cloud ice — a quick glance at Figure 4.10 indicates that it should be present in $S_{v,ci}$ and $S_{cl,ci}$ but not $S_{ci,s}$ because this is an *outward* flowing parameter from cloud ice. A number of constants are also present in the equations, which are denoted K with a subscript that is relevant to their function.

The definition of the actual equations of a water model is generally a fairly flexible affair, due to the sheer complexities of the physical processes that are being approximated (within meteorological literature they vary widely and sometimes can appear to be treated more as implementation issues). One of the aims of this thesis is to provide a foundation for future work, so experimentation with the water model is actively encouraged. For those interested, Houze [1993] provides a good introduction to the subject of water continuity models.

4.2.4.1 Condensation and Deposition: $S_{v,cl}$ and $S_{v,ci}$

The transfers of vapour to cloud liquid and cloud ice are extremely similar, so we shall deal with them together. They are essentially dependent on the largely misunderstood concept of 'saturation' which is actually an equilibrium point when, in the case of gas and liquid, the rates of continual condensation and evaporation become equal. (For more information on this topic the reader is referred to

Bohren and Albrecht [1998] who dedicate several pages to its discussion). Once a parcel passes beyond this equilibrium point the rate of condensation becomes greater than the rate of evaporation and the net result is one of condensation.

The initial process of condensation (gas to liquid) or deposition (gas to solid) when there is no liquid or solid water present is known as *nucleation*. Water is actually very reluctant to undergo this state transformation in a pure environment and the process is highly dependent on impurities in the atmosphere which take the form of small *nucleation particles*. Until nucleation occurs, the parcel is likely to undergo *supersaturation*, where more water vapour is present than should be according to the calculated saturation level. Once an initial quantity of water has condensed on the nucleation particles, the process becomes easier and the amount of vapour falls to the saturation level. For the transfer between vapour and cloud liquid, in the initial case when no cloud liquid is present, this is expressed as:

$$S_{v,cl} = C_{cl}\sigma\left(q_v - q_{vsl}K_{sl}\right)$$

when $q_{cl} = 0$, and where q_{vsl} is the liquid saturation vapour mixing ratio, σ is a temperature control parameter and K_{sl} is the liquid supersaturation control parameter. The value of σ smoothly varies between 0 and 1 as the temperature rises from 253.15K to 273.15K (-20°C to 0°C), to reflect the counter intuitive observation that temperatures must be well below freezing before cloud ice forms from deposition [Schultz 1995]. The supersaturation constant K_{sl} may be adjusted and was set at 1.2 for this study as found after a period of experimentation. The visual effect of the supersaturation is for the cloud to initially form at a slightly higher altitude than the specified cloud base, before spreading downwards to meet this level.

Beyond nucleation, this simplifies to:

$$S_{v,cl} = C_{cl}\sigma\left(q_v - q_{vsl}\right)$$

The liquid saturation vapour mixing ratio is defined as:

$$q_{vsl} = \frac{0.622 e_{sl}}{p - e_{sl}}$$

where p is the pressure of the parcel, and e_{sl} is the liquid saturation vapour pressure, which is a partial pressure that can be calculated using an empirical approximation: [Bolton 1980; Rogers and Yau 1989, p. 16]

$$e_{sl} \approx 0.6112 \exp\left(\frac{17.67T}{T + 243.5}\right)$$

where T is the temperature of the parcel.

Schultz uses an exponential formulation to describe ice particle nucleation but it was decided that a similar mechanism to that for cloud water should be used here to aid clarity and usability rather than accuracy. For the transfer between vapour and cloud ice, in the initial case when no cloud ice is present:

$$S_{v,ci} = C_{ci}\sigma' \left(q_v - q_{vsi}K_{si}\right)$$

when $q_{ci} = 0$, and where q_{vsi} is the ice saturation vapour mixing ratio, σ' is the inverse temperature control parameter (i.e. $\sigma' = 1 - \sigma$) and K_{si} is the ice supersaturation control parameter. Again, beyond nucleation this simplifies to:

$$S_{v,ci} = C_{ci}\sigma' \left(q_v - q_{vsi}\right)$$

The ice saturation vapour mixing ratio is similarly defined as:

$$q_{vsi} = \frac{0.622e_{si}}{p - e_{si}}$$

The ice saturation vapour pressure e_{si} is then estimated using Buck's empirical method, which is optimised for temperatures between 223.15K to 273.15K (-50°C to 0°C): [1981]

$$e_{si} \approx 0.6112 \exp\left(\frac{22.452T}{T + 272.55}\right)$$

The reverse processes of evaporation (liquid to gas) and sublimation (solid to gas) are covered later in Section 4.2.4.6 on page 90.

4.2.4.2 The Bergeron Effect: $S_{cl,ci}$

Following Schultz, this scheme does not implement the freezing of cloud water to cloud ice, which does not actually account for the majority of the water transferred between these states. Instead, a process known as the *Bergeron effect* is modelled. This is dependent on the fact that the saturation vapour pressure for liquid is greater than that for ice. Following Storelvmo et al. [2008] there are three possible states for a parcel to be in, with respect to its vapour pressure e_v:

1. $e_v > e_{sl} > e_{si}$, i.e. there is a lot of water vapour. This results in both ice and liquid droplet growth.

2. $e_{sl} > e_v > e_{si}$, i.e. liquid should be evaporating, but ice should be forming. This is the Bergeron effect.

3. $e_{sl} > e_{si} > e_v$, i.e. there is not much water vapour present. This results in both ice and droplet shrinkage.

When the required conditions are satisfied, the transfer due to the Bergeron effect is expressed by:

$$S_{cl,ci} = C_{ci}\left(q_{cl} - q_{vsl}\right)$$

4.2.4.3 Snow and Rain: $S_{cl,r}$ and $S_{ci,s}$

For snow and rain, the transfers are assumed to start occurring when the corresponding cloud water level rises above a threshold value. Beyond this value, cloud particles are assumed to have grown to the point where they are affected by gravity and fall from the cloud. For snow, this is expressed as:

$$S_{ci,s} = C_s\left(q_{ci} - q_{isThresh}\right) K_{cs}\Delta t$$

where $q_{isThresh}$ is the controllable cloud ice to snow threshold, set to the value of $0.0001\,kg.kg^{-1}$ in this study, and K_{cs} is the snow conversion rate set at 0.1 here, both as found through experimentation. Similarly for rain:

$$S_{cl,r} = C_r\left(q_{cl} - q_{lrThresh}\right) K_{cr}\Delta t$$

where $q_{lrThresh}$ is the controllable cloud liquid to rain threshold, set to the value of $0.0007\,kg.kg^{-1}$ in this study, and K_{cr} is the rain conversion rate, again set at 0.1 here (as found through experimentation). When $q_r > 0$, the mechanism of *collection* occurs, in which rain literally (and rapidly) collects additional cloud liquid. This equation then becomes:

$$S_{cl,r} = C_r q_{cl} q_r K_{cr}\Delta t$$

Visually, adjusting the threshold values for snow and rain controls how large the cloud is before it starts precipitating.

4.2.4.4 Hail: $S_{cl,h}$, $S_{s,h}$ and $S_{r,h}$

Three processes are covered here, the first of which is growth from cloud liquid due to the *riming* of graupel and is treated in the same way as rain collection:

$$S_{cl,h} = C_h q_{cl} q_h K_{rh}\Delta t$$

where K_{rh} is the hail riming rate, which was set at 0.1 (as found through experimentation). The second of these processes is the riming of snow, where snow collects cloud liquid and becomes graupel.

$$S_{s,h} = S_{cl,h} = C_h \min\left(q_{cl}, q_s\right) K_{rs}\Delta t$$

where K_{rs} is the snow riming rate, again set to 0.1 (again, as found through experimentation). With this transfer, it is important to remember that equal measures of cloud liquid and snow are required. The third process is the freezing of rain to become hail, and Schultz's equation is used as a basis here:

$$S_{r,h} = C_h \left(265.0 - T\right)^2 K_{fr} \Delta t$$

where $T > 265K$ and K_{fr} is the rain freezing rate, set to 0.1 as found through experimentation.

4.2.4.5 Melting: $S_{ci,cl}$, $S_{s,r}$ and $S_{h,r}$

The transfer from cloud ice to cloud liquid is due to melting, and takes the form of:

$$S_{ci,cl} = C_{cl} \delta q_{ci}$$

Where δ is a temperature control parameter which is 1 if the parcel's temperature is above freezing (273.15K), and 0 otherwise. In this model, snow melts to rain according to the similar equation:

$$S_{s,r} = C_r \delta q_s$$

and finally, hail melts to rain according to:

$$S_{h,r} = C_r \delta q_h K_{mh} \Delta t$$

where K_{mh} is the hail melting rate, set to 0.01. This rate can have quite an effect on precipitation, since most rain actually starts as snow which becomes hail before melting into rain. By setting this conversion rate low, we therefore get more hail at ground-level.

4.2.4.6 Evaporation and Sublimation: $S_{cl,v}$, $S_{ci,v}$, $S_{r,v}$, $S_{s,v}$ and $S_{h,v}$

These transfers occur when the level of vapour present is less than the saturation levels of liquid or ice water. Following Schultz's lead again, the treatment here reflects the facts that "small particles evaporate before large particles and liquid evaporates before ice" (due to the lower saturation vapour pressure) [1995]. This is reflected by the order in which the deficit beyond the saturation point is 'repaid': first is cloud liquid, then rain, cloud ice, snow and finally hail. The equations, in the correct order, are:

$$\begin{aligned}
S_{cl,v} &= C_v\left(q_{vsl} - q_v\right) K_{el}\Delta t \\
S_{r,v} &= C_v\left(q_{vsl} - q_v\right) K_{er}\Delta t \\
S_{ci,v} &= C_v\left(q_{vsi} - q_v\right) K_{ei}\Delta t \\
S_{s,v} &= C_v\left(q_{vsi} - q_v\right) K_{es}\Delta t \\
S_{h,v} &= C_v\left(q_{vsi} - q_v\right) K_{eh}\Delta t
\end{aligned}$$

where the K parameters are the evaporation rates which, after a period of experimentation, were set as: $K_{el} = 0.03$, $K_{er} = 0.005$, $K_{ei} = 0.02$, $K_{es} = 0.0025$, $K_{eh} = 0.0008$. Thus, if the deficit is greater than the amount of cloud liquid present, then rain will evaporate; if the vapour level is still below the saturation point, then cloud ice will sublimate, and so on. These are the only transfers in which the order of evaluation is of real importance. The conversion rates also play a large role in the appearance of the cloud formation. The precipitating transfers above do not occur instantaneously because otherwise the precipitates would not actually reach the ground. Care must be taken with the time-steps of the simulation with regard to the cloud water and cloud ice evaporation rates. If the simulation time-step is greater than the conversion rates, it is possible that small amounts of cloud can condense and (seemingly) rapidly evaporate the very next frame, which can visually take the form of random noise.

4.2.4.7 Heating Effects

Phase transitions result in either heating or cooling of the parcel, so the total mass of water transferring between states is used to change the parcel's temperature:

$$\Delta T = L_c q_{condensation} + L_d q_{deposition} + L_f q_{freezing}$$

where L_c, L_d and L_f are the latent heats of condensation, deposition and freezing and the mixing ratios are the amounts of water that are being transferred during the timestep. Schultz limits certain transfers to prevent excessive heating or cooling of the parcel, but it was felt that this might also limit the effectiveness of the user controls and so was not included in this scheme.

4.2.5 Summary of The Cloud Model

Our model started with a static background atmosphere and parcels of air that move through it and eventually form clouds. A parcel's pressure equalises to its surroundings and each parcel has its own temperature. The parcel's buoyancy is dependent on the difference between its temperature and the background atmospheric temperature. This means we can influence the parcel's vertical movements by choosing the parcel's initial temperature and controlling the background atmospheric temperature profile. We use a tool based on a meteorological chart to choose heights for the cloud base and cloud top, then the tool generates the background atmospheric temperature profile, parcel source temperature and vapour levels that will generate this cloud. Parcels contain six categories of water: vapour, cloud liquid, cloud ice, rain, snow and hail. A set of water continuity equations define how water changes from one category to another. To help further direct our clouds, we put control parameters into these equations to enhance or restrict water flow between categories. Finally, when water state changes occur they result in either additional warming or cooling of the parcel, which further affects buoyancy. The result of our cloud simulation is a set of water densities for each of the water categories, which we must now visualise.

4.3 Rendering

Although the primary concern of this work is not rendering, visual output must be obtained to validate the results of the simulation. The method that was used is a simple GPU ray-marcher with 'interactive' and 'offline' quality modes defined by simply adjusting the number of samples in the primary and secondary ray integrations.

The simulation output is effectively that of a multi-field weather simulation — a three dimensional scalar field of water mixing ratios for multiple water categories. As described in Chapter 3, single-scattering for multi-field weather data is well defined [Riley et al. 2004b] and this basic technique was used.

At each step, the ray marcher must determine the colour and alpha contributions of the sample. The ray marcher calculates the lighting intensity along a view ray, $\vec{v}$, between the near and far bounds of the cloud volume by evaluating the following integral:

$$L(\vec{v}) = \int_{near}^{far} L_{SS}(\vec{p}) L_{MS}(\vec{p}) d\vec{p}$$

where L_{SS} is the single scattering intensity and L_{MS} is the multiple scattering intensity at position $\vec{p}$. To build the single scattering intensity, we first need to define some terms.

The area of a particle from a specific water category (sub or superscript 'field') that can scatter light is the scattering cross section which may be defined as twice the geometric cross section:

$$\sigma_{sc}^{field} = 2\pi R_{field}^2$$

The number of particles per cubic metre, η, is then defined by:

Category:	Value:	Approx Particle Radius:
Cloud Liquid	150	0.01mm
Cloud Ice	23	0.05mm
Rain	0.11	1mm
Snow	9.1	2mm
Hail	0.66	2.5mm

Table 4.1: Values for the k^{field} constant, arrived at by calculating initial values from approximate average droplet sizes and then manually adjusting by comparing rendered outputs with photographic references.

$$\eta^{field} = \frac{\rho_{air} q_{field}}{M^{field}}$$

where q is the mixing ratio for that category and M is the mass of one particle. The probability per unit length of hitting a particle is the scattering coefficient which may then be defined as:

$$\beta_{sc}^{field} = \sigma_{sc}^{field} \eta$$

To assist with the implementation, the constants were grouped together to give:

$$\beta_{sc}^{field} = k^{field} \rho_{air} q_{field}$$

where:

$$k^{field} = \frac{\sigma_{sc}^{field}}{M^{field}}$$

with values as shown in Table 4.1. The constant k was found to be more intuitive as a rendering parameter, since higher values directly correspond to a denser cloud.

The sample's scattering coefficients are then combined into the effective scattering coefficient:

$$\beta_{sc} = \sum_{\text{All Fields}} \beta_{sc}^{field}$$

The extinction coefficient is the sum of the scattering coefficient and the absorption coefficient. The albedo is the ratio between the scattering and absorption coefficients. Clouds are highly scattering so we assume the albedo is close to 1, therefore we neglect absorption and the extinction coefficient becomes the scattering coefficient.

The transparency of a sample between positions is found by integrating the effective extinction coefficient:

$$T(\vec{p_1}, \vec{p_2}) = e^{-\int_{\vec{p_1}}^{\vec{p_2}} \beta_{ex} ds}$$

The final alpha channel output of the ray marcher along a viewing ray is $1 - T(\vec{p_{near}}, \vec{p_{far}})$.

The result when light is scattered by a particle is defined by the phase function and different particles give different phase functions. We thus will use a different phase function for each water category. This demonstrates the advantages of our cloud model by allowing category specific optical effects such as rainbows in the rain water, haloes in the cloud ice, coronas in cloud water and glories in cloud ice and water. We define the phase function with respect to the angle between the sun ray and the viewing direction, which is sufficient for most cases but will not allow the more complex (and much rarer) ice haloes to be viewed. A phase function, $P_{field}(\theta)$, is defined for each water category and then the results from all the categories are combined to form the effective phase function:

$$P(\theta) = \frac{\sum_{\text{All Fields}} \beta_{ex}^{field} P_{field}(\theta)}{\beta_{ex}}$$

The individual phase functions were initially generated by the MiePlot software [Laven ca.2012], converted to floating point bitmaps and adjusted to achieve the desired visual appearance.

The single scattering intensity can now be defined as:

$$L_{SS}(\vec{p}) = T(\vec{p}, \vec{d_{sun}}) P(\theta) L_{sun} \beta_{ex}(\vec{p})$$

where $\vec{d_{sun}}$ is the point at the boundary of the simulation volume in the direction of the sun when starting from point $\vec{p}$ and L_{sun} is the intensity of the sun.

Clouds are highly scattering media, so multiple-scattering plays an important role in their appearance. As described in Chapter 3, there have been numerous attempts at simulating the multiple-scattering properties of participating media, which vary in complexity and computational expense. For this work, a simple technique was chosen, based on colour gradients. For a sample position, rays are shot towards the sun direction, ground direction and straight up (to emulate sky direction). The obtained optical densities, which are combined across all water categories, are used as interpolants on the colour gradients to determine a set of multiple-scattering contributions. This method was chosen for its computational simplicity, its ability to be artist-directed and because the advantages of the multi-field water model are best demonstrated by the single-scattering contribution, due to the more isotropic nature of multiple-scattering interactions — while not

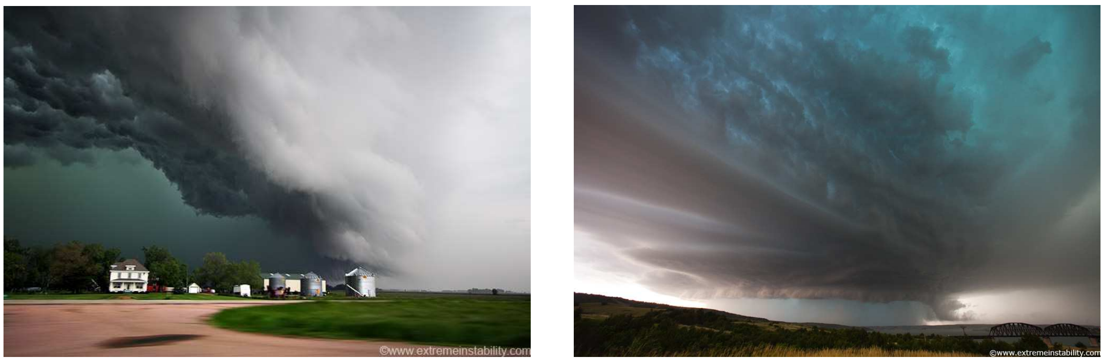


Figure 4.11: The ominous green tint to these large stormcells is due to high levels of ice in the cloud. The simulation output will indicate the location of ice but the rendering of this effect is left for future work. Images credited to Mike Hollingshead [Hollingshead ca.2009].

purely isotropic, multiple-scattering tends to 'blur' out the high frequency details of the phase function [Bouthors et al. 2008]. One limitation of this approximation is that it prevents the 'green tint' that can sometimes be seen in cloud that is heavily laden with ice (see Figure 4.11). Such effects are left as future work.

The high density of the cloud water means that viewing rays often do not travel very far into a cloud. This will result in banding artefacts where the rays' first samples of a cloud are at different depths. A method of visually reducing this is to offset the sampling distances by random values. Although this gives a noisy result, it is visually more acceptable than banding. More random accesses to a volume texture results in poor caching, however, so rendering performance suffers. A similar approach to reduce banding is to quickly reject a sample that is too opaque and try again with a shorter sampling distance. This results in more samples, but can improve the visual result. Artefacts also appear when the sampling distance is too great, so that some samples hit the cloud while others do not. This issue is clearly visible with the fast 'interactive' rendering mode.

4.4 Implementation

In this section we cover how to integrate the water model into a full Eulerian-based system and provide details of the prototype implementation.

4.4.1 System Design

The first issue to consider is that of data storage. Each grid-cell needs to store a potential-temperature and six mixing ratios, one for each of the water categories.

During each simulation update cycle the cloud model generates a buoyancy value for each grid-cell which must also be stored so that it can be passed to the fluid system. The initial conditions of the volume data are for the temperature to be set to that corresponding with the background atmospheric temperature, while the water values are all set to zero. A one-dimensional array is required to store the background atmospheric temperature profile, which is needed in the water-update and buoyancy equations. The water control parameters could each be global variables, though more interesting results can be obtained if they are each 2D textures that map to the horizontal plane and this is what was implemented in the prototype system. Wind and turbulence volumes also need to be stored, though these can be at much lower-resolutions than the main simulation volume. All these properties are in addition to those required by the fluid-dynamics solution and rendering technique.

The order of operations for a generic system, including fluid-dynamics, is then to loop over the following steps:

- Advect the water properties using the fluid velocity field output from the fluid solution as calculated in the previous iteration (the velocity field is usually initialised to zero)
- Seed new water and temperature at ground-level according to source conditions
- Evaluate the water continuity equations
- Work out any heating effects resulting from state changes
- Store the new water and temperature levels
- Calculate and store the buoyancy force
- Advect the fluid velocities, taking into account boundary conditions
- Add additional forces to the fluid (wind, turbulence and buoyancy)
- Enforce the divergence-free condition (e.g. using Hodge-Helmholtz decomposition, solving the resulting Poisson equation with Jacobi iteration to create a scalar ‘pressure’ field and then subtracting its gradient from the velocity field)
- Store the resulting divergence-free fluid velocity field for use in the next iteration
- Render the water categories (excluding vapour)

This scheme was used as the basis for the prototype system, the implementation of which we shall now examine.

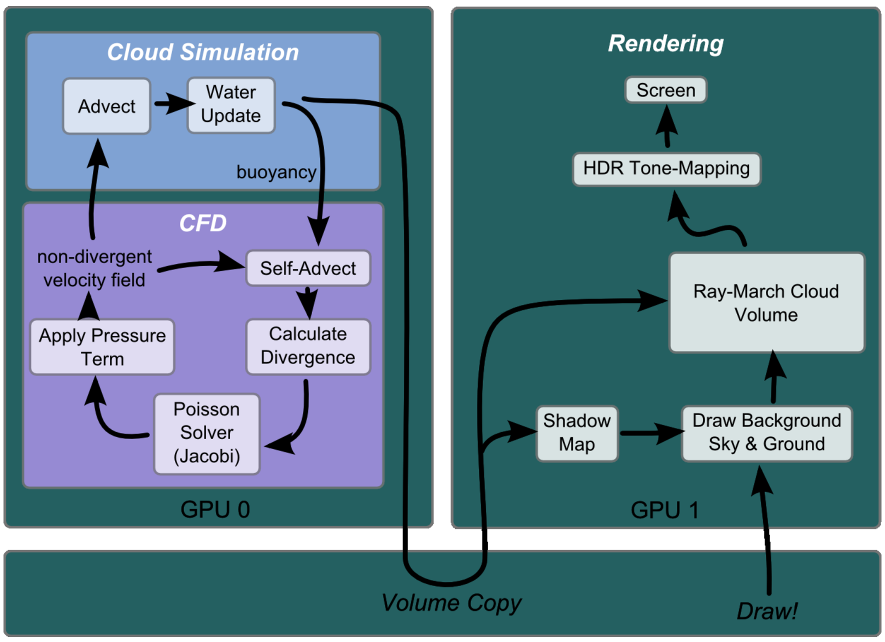


Figure 4.12: One GPU is used for the cloud and fluid simulation, while the other asynchronously renders the cloud volume.

4.4.2 System Implementation

The final prototype system was designed to run at near interactive-rates, with a higher quality 'snapshot view', and to make heavy use of the GPU. It was implemented in CUDA and tailored to the development platform, which contained two CUDA devices. An overview of the system architecture is shown in Figure 4.12. Within the prototype the simulation updates are decoupled from the rendering updates and each task is performed on a separate GPU. After each simulation update, the simulation GPU makes a copy of the water categories which are then transferred to the rendering GPU. The simulation volumes are all stored in 32-bit floating-point format, with any less precision being found to adversely affect the stability of the simulation. However, it was decided that this accuracy was not required for the rendering technique so the copy process also downgrades the data to an 8-bit format. This dramatically reduces the amount of data that needs to be transferred between GPUs and greatly improves the rendering performance. The one-dimensional background atmospheric data is also stored in 32-bit floating-point format but the 3D wind and turbulence and 2D control parameters are only 8-bit.

The simulation domain was set to $30km$ on each horizontal axis and $10km$ vertically, with results obtained at different resolutions (as described in the next section). The height matches that of an idealised troposphere and the chosen horizontal sizes are small enough to give a reasonable resolution within the available memory while being just about large enough for a sizable, if stationary, cumulonimbus to form. The fluid-dynamics solution is based on that by Stam [1999], with the addition of a staggered MAC grid in which velocities are stored at the cell boundaries rather than the cell centres [Bridson 2008]. This method assists with evaluating spatial derivatives across the cells and because we do not need to calculate such derivatives for all other cell properties (such as water categories and temperature) these are stored at cell centres (the reader is directed to Bridson [2008] for advice on correct data storage within a staggered MAC-grid arrangement). The boundaries enforce a no-stick condition at the sides and top, while a no-slip condition is enforced at the ground.

Within the water advection step, the vapour and cloud water categories are moved according to the input advection velocity while the precipitating categories also attempt to fall to the ground. Each precipitation state falls at a constant rate, with rain assumed to fall at a rate of $10ms^{-1}$, snow at a rate of $3ms^{-1}$ and hail at a rate of $25ms^{-1}$. These values were inspired by average rates (see Section 2.4.1.1 on page 37) and then adjusted after a period of experimentation.

Basic wind control was provided for in the prototype system with a simple editor. The user can place wind forces within the volume, which are then built into a single volume (using simple summation of the wind vectors) and sent to the fluid advection step. Here, a force is applied in the wind's direction until the wind velocity is reached. A turbulence is also added at this point as an additional force which helps to provide an uneven appearance to the clouds. This turbulence is a divergence-free vector field generated from the gradient of a random vector field $32 \times 32 \times 32$ voxels in size with periodic boundaries (this uses the vector field identity that the divergence of a gradient is zero).

A skew-T editor was provided for the user to adjust the temperature profile and a simple 'painting' editor for the water control parameters. The output of each of these are transferred to the simulation GPU. A change to the temperature profile triggers a reset of the simulation, since it must change the initial conditions and too large a change can result in fluid simulation instabilities. Adjustment of the water control parameters may be performed in the prototype without resetting the simulation. Eight control parameters are provided (and stored in two 4-channel, 8-bit-per-channel, textures), six of which are for the water control parameters, one for the source temperature and one for the source vapour level. These source parameters are used by the simulation at ground level to modulate between the background atmospheric temperature (and zero vapour) and the values provided by the user through the skew-T editor tool. Example use of the skew-T editor is show in the next section, Figure 4.23 on page 111.

The cloud rendering technique was described in Section 4.3. The results from the CUDA renderer are blended with the background scene using alpha blending, with the alpha channel computed according to the total eye-ray density of the water categories. In the prototype this background consists of a sky-dome with a gradient texture applied and a ground-dome (with a radius matching that of the Earth) displaying Landsat 7 satellite imagery [NASA Landsat Program 2008]. The entire rendering pipeline is HDR and a tone-mapping and bloom are applied as post-processes, with an exposure parameter manually adjusted by the user.

4.5 Results

4.5.1 Performance

The development platform was a quad-core 3GHz AMD-based workstation with one NVIDIA GeForce 560 Ti card and one GeForce 480, with the rendering performed on the former and the simulation on the latter. Computational times are shown in Table 4.2 and indicate that the evaluation of the water equations is greatly overshadowed by the overall computational expense of the fluid-dynamics solution. None of the methods are heavily optimised so it is expected that performance could be improved, especially of the fluid dynamics solution.

Resolution	Water	Dyn.	Sim.
$96 \times 32 \times 96$	2.8 ms	60.6 ms	4.2 UPS
$192 \times 64 \times 192$	5.1 ms	110.5 ms	2.2 UPS
$240 \times 80 \times 240$	6.3 ms	230.5 ms	1.5 UPS

Table 4.2: Average computational times for the water simulation, fluid-dynamics simulation and the complete simulation update rate (in Updates Per Second). Results are averaged over 100 simulation updates.

4.5.2 Visual Results

The difference between the fast and slow rendering methods are shown in Figure 4.13. The quality is significantly reduced for the fast method, but interaction with the system is possible. Different sampling strategies are shown in Figure 4.14. In the simplest method, the hardware linear interpolation technique is used. This results in good frame-rates but gives a blocky appearance in regions of high density contrast, which is typical for heavily water-laden cumuliform clouds. A solution is to use an alternative to linear interpolation of the 3D texture, such as cubic interpolation, but such methods are not currently hardware accelerated and so the increase in visual quality comes at the price of a performance decrease. Finally, procedural noise is added to emulate higher frequency detail than is present in the simulation. The results are not a perfect match to real clouds, but are a significant improvement on plain linear interpolation.

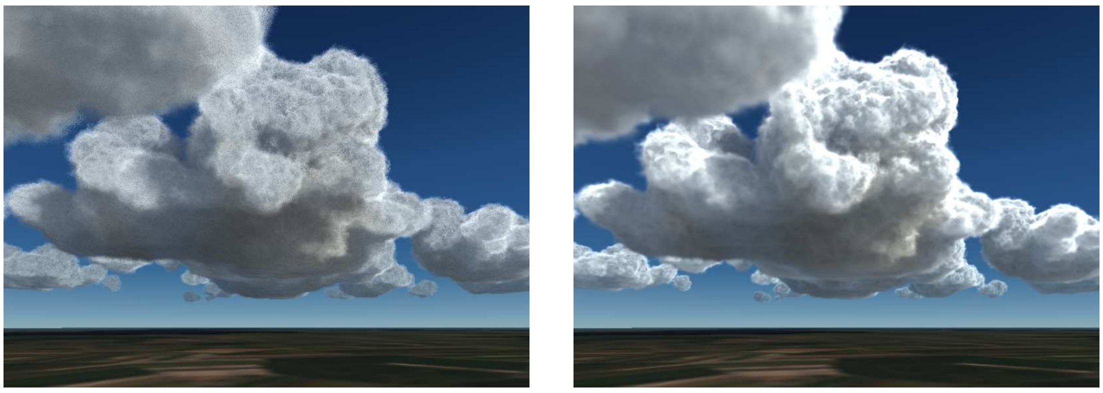


Figure 4.13: The same cloudscape with fast (left) and slow (right) rendering techniques. The difference is the number of samples in the primary and secondary rays. The fast method was rendering at approximately 0.6 frames-per-second, while slow images typically take around 5 seconds each to generate.

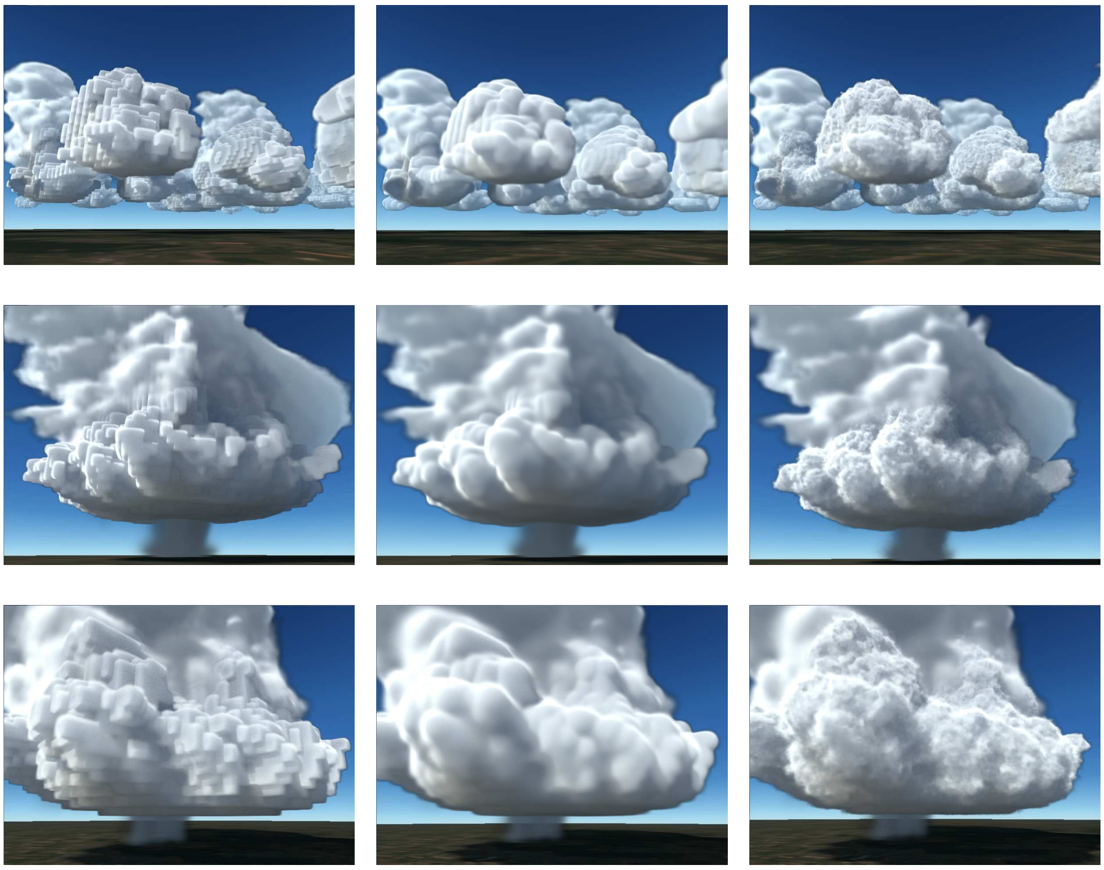


Figure 4.14: The same cloudscapes with different sampling methods. Left: linear interpolation, notice the blocky appearance of denser clouds, showing the actual resolution of the cloud simulation output. Middle: simple cubic interpolation removes the blocks but gives a too-smooth appearance. Right: adding procedural noise to cloud water and ice categories gives a sense of high resolution detail.

Previous works in computer graphics have limited cloud simulation to just vapour and cloud water categories. This means they cannot simulate cumulonimbus clouds, which precipitate by definition. The water model presented in this work includes cold water and precipitation categories, and so can be used to simulate clouds of this type. A comparison is shown in Figure 4.15 which demonstrates the visual differences for a large cumuliform cloud that should be precipitating and include high-altitude ice formations. Precipitation details are also shown in Figure 4.16.

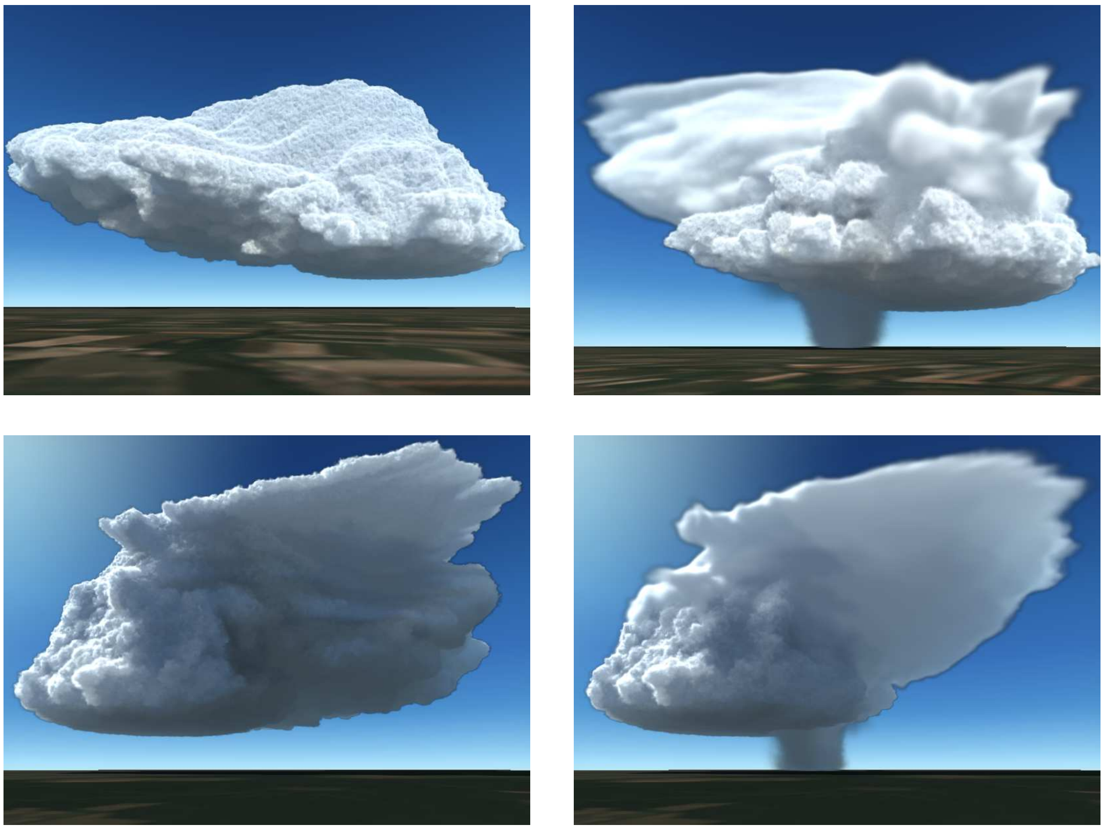


Figure 4.15: Two angles of the same cloud formations. Left: previous works were limited to water in only the vapour and cloud liquid categories and can only result in large cumulus formations. Right: the work presented here allows for cumulonimbus simulation. Notice the softer appearance of upper parts of the cloud where ice and snow are prevalent, and the region of precipitation.

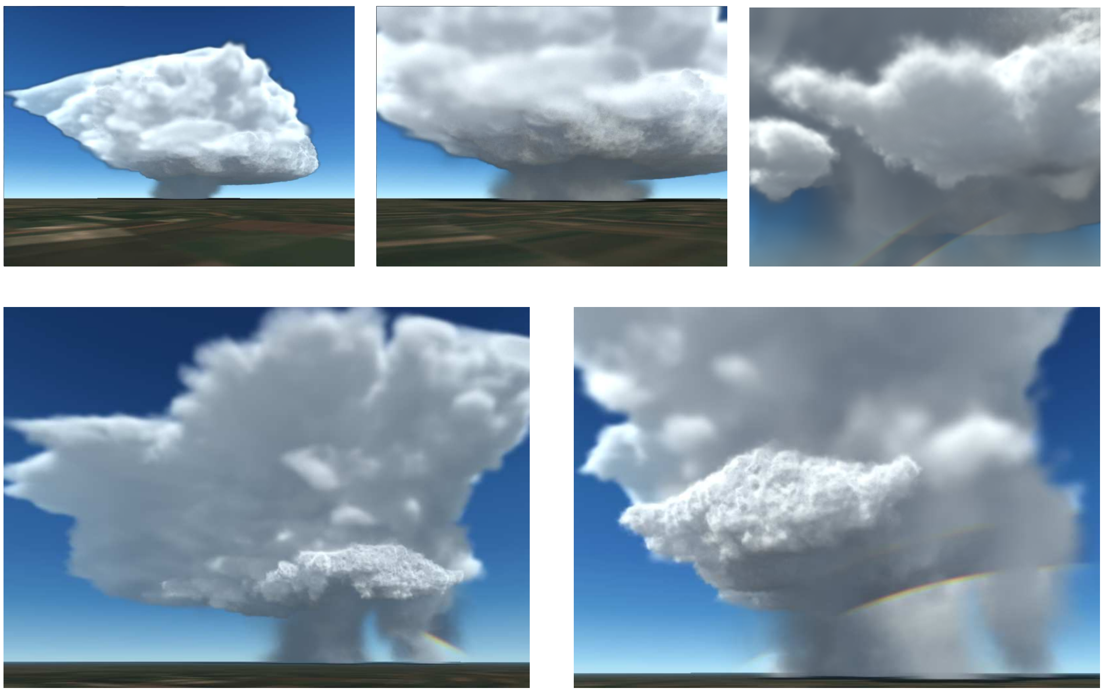


Figure 4.16: Simulation of precipitation from large clouds. Note the double rainbow.

Another advantage of a six-state water model is the ability to visually represent the separate water categories differently. Ice and water categories exhibit different optical effects, ranging from halos to rainbows, some of which were used in the rendering technique as indicated in Table 4.3. Visual results of these are shown in Figure 4.17.

Water Category	Corona	Halo	Rainbow	Glory
Cloud Liquid	x			x
Cloud Ice		x		x
Rain			x	
Snow		x		
Hail			x	

Table 4.3: The optical effects used in the prototype system, which are exhibited in the water phase functions.

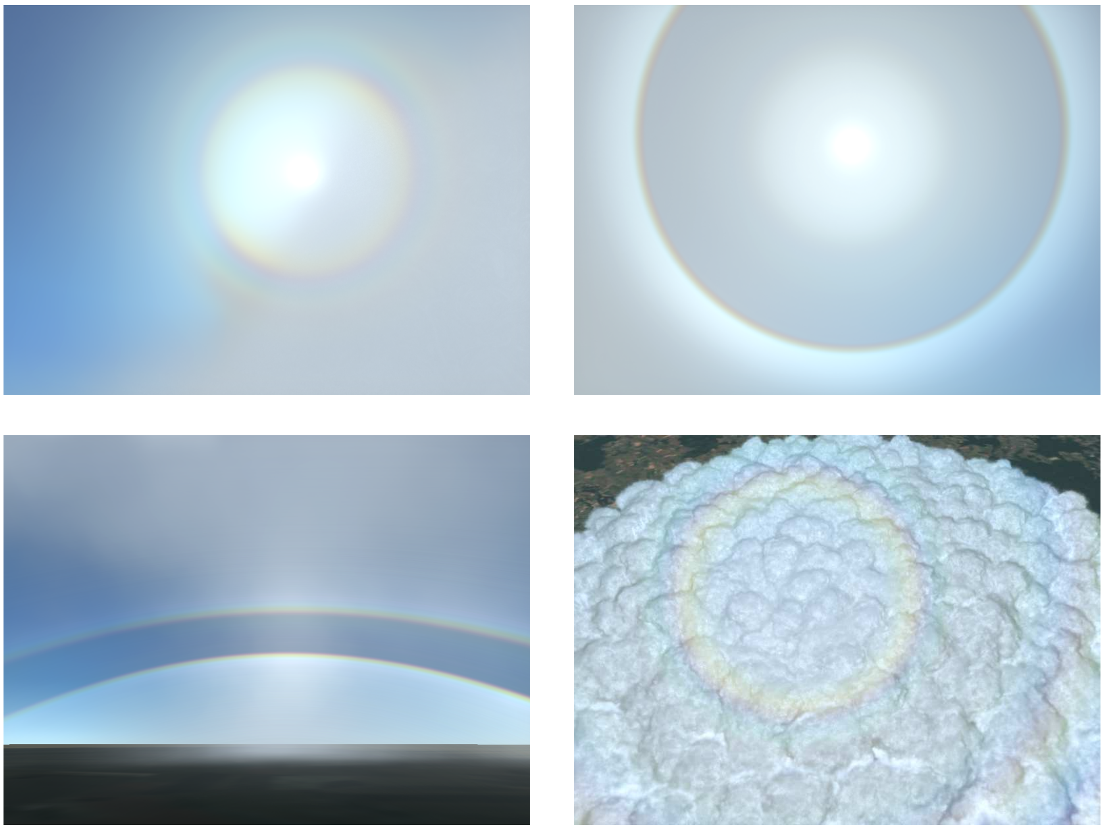


Figure 4.17: Optical effects in the simulation are a result of the different phase functions which are used for the different water categories. Top-left, corona around the sun as seen through cloud water; top-right, 22 degree halo seen through a layer of ice and snow; lower-left, rainbow; lower-right, glory opposite the sun on a layer of cloud.

4.5.2.1 Visual Comparisons

The purpose of this section is to demonstrate that the simulation system can produce similar cloud forms to those found in nature. Figure 4.18 shows various cumulonimbus formations while Figure 4.19 shows rain detail. Figure 4.20 shows ‘anvil’ clouds, which form when a deeply convecting cloud rises to a stable region of the atmosphere and spreads out. Cloud details are compared in Figure 4.21 and in Figure 4.22 a comparison is made with regions of different texture.

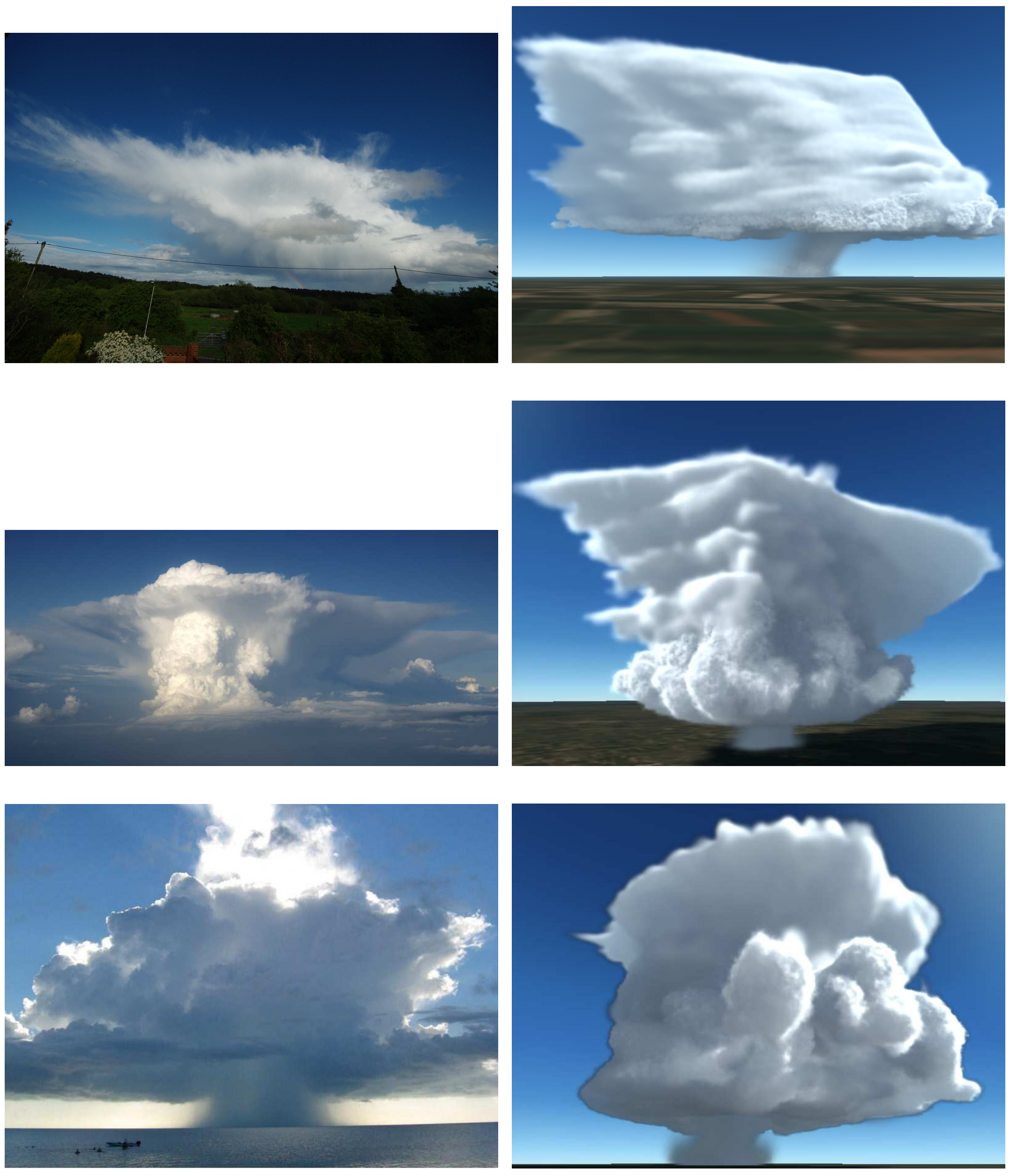


Figure 4.18: Left, photographs of classic cumulonimbus formations. Right, simulations. In the top pair, the wind makes the clouds appear to lean to the left. Photograph credits: top left, the author April 2009; middle left, Mark McCaughrean via The Cloud Appreciation Society gallery 2011 and bottom left, Dene Georgelin via The Cloud Appreciation Society gallery ('cloud of the month', May 2008) [Pretor-Pinney ca.2012].

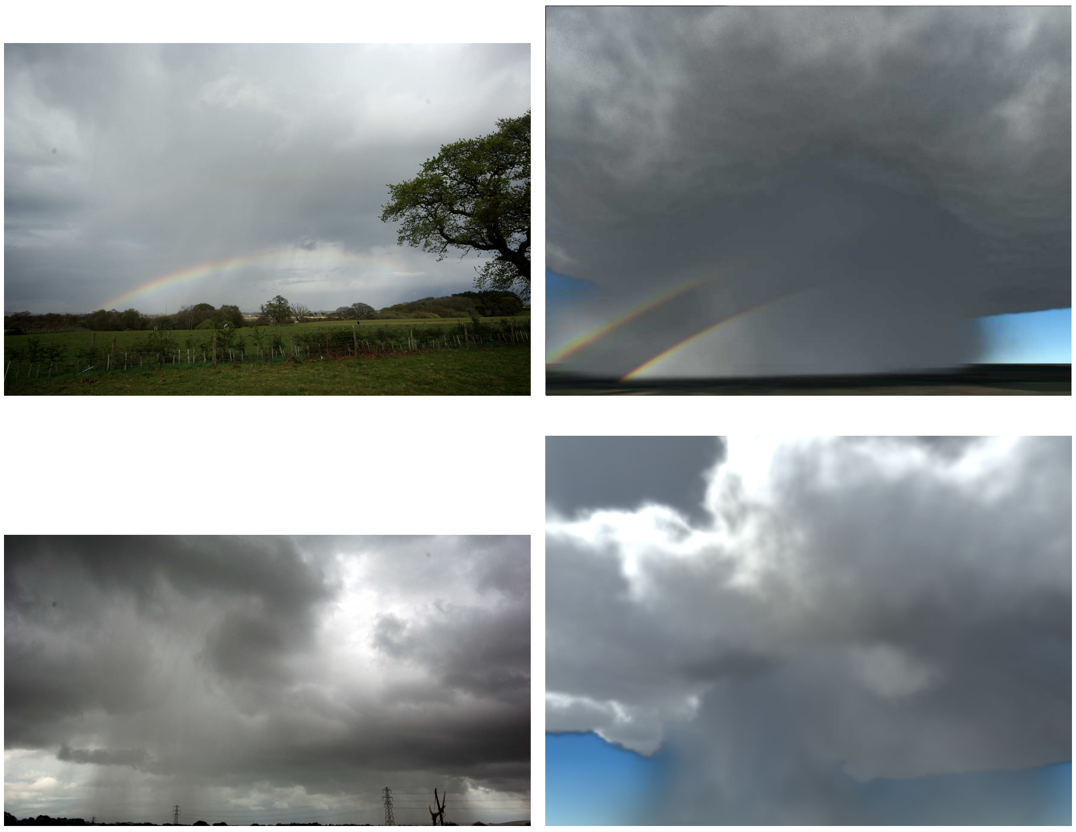


Figure 4.19: Left, photograph of rain. Right, simulation. Note the double rainbow in the top pair. Photographs by the author, April 2012.

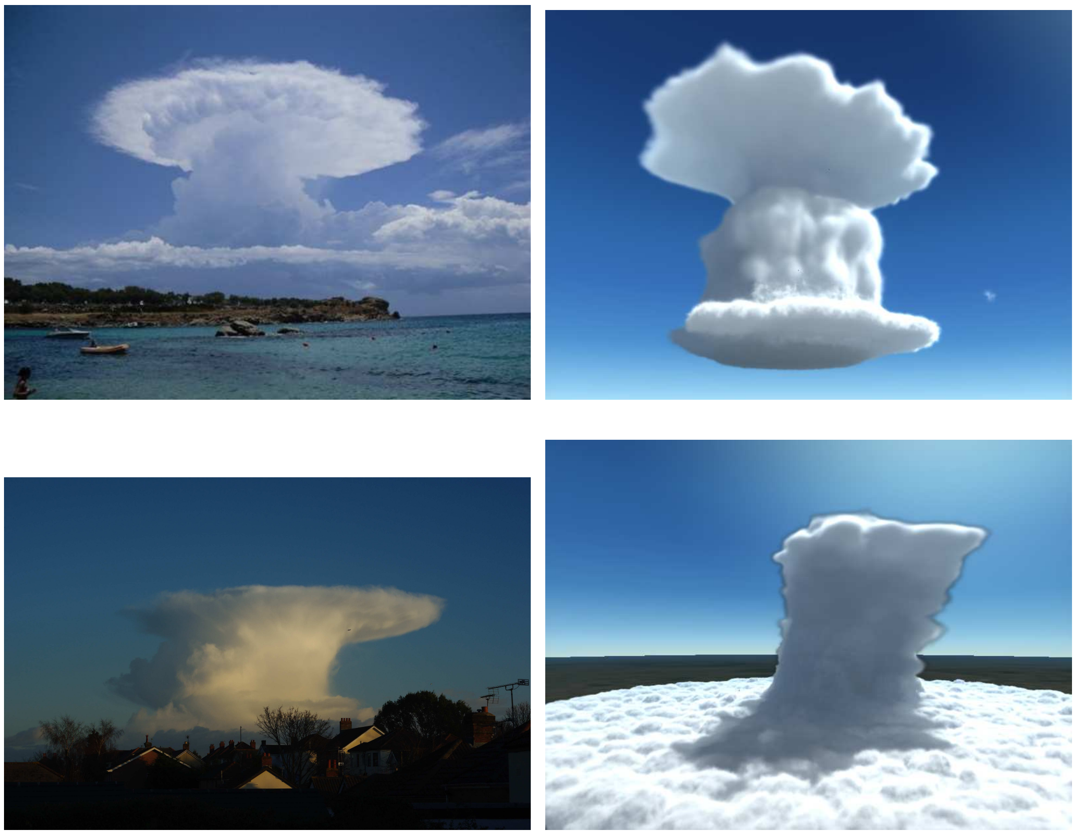


Figure 4.20: Left, photographs of ‘anvil’ clouds. Right, simulations. Rising cloud hits a region of high stability and can travel no further vertically, so it spreads out horizontally. Photograph credits: top left Damian Cox via The Cloud Appreciation Society gallery 2009 [Pretor-Pinney ca.2012]; lower left by the author, December 2011.

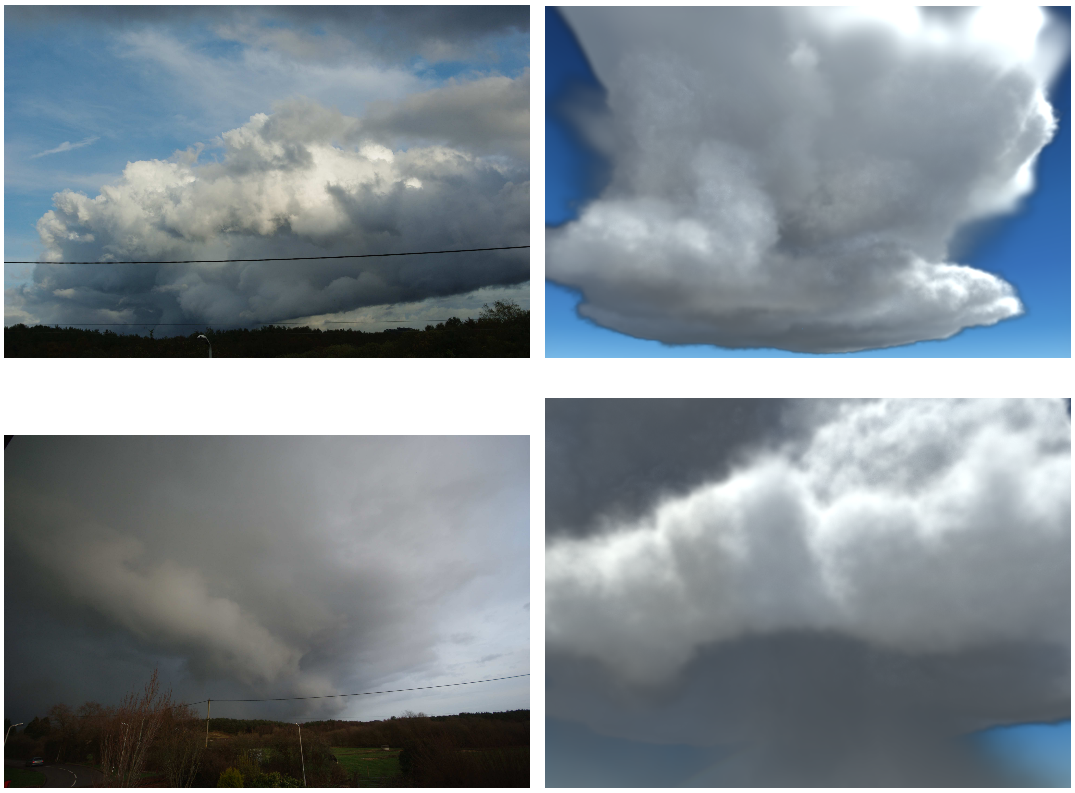


Figure 4.21: Left, photographs. Right, simulations. The lower pair show the edges of heavy rain clouds. Photographs by the author.

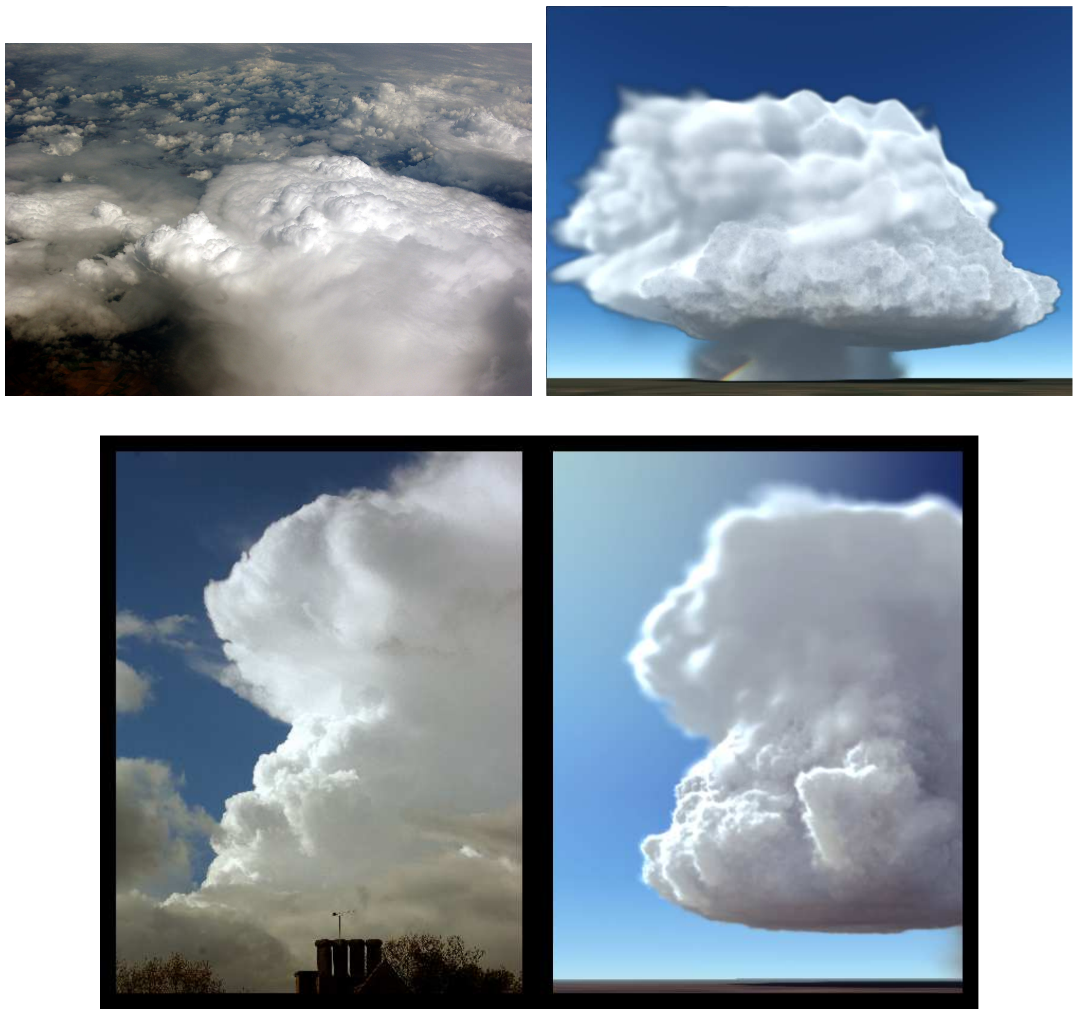


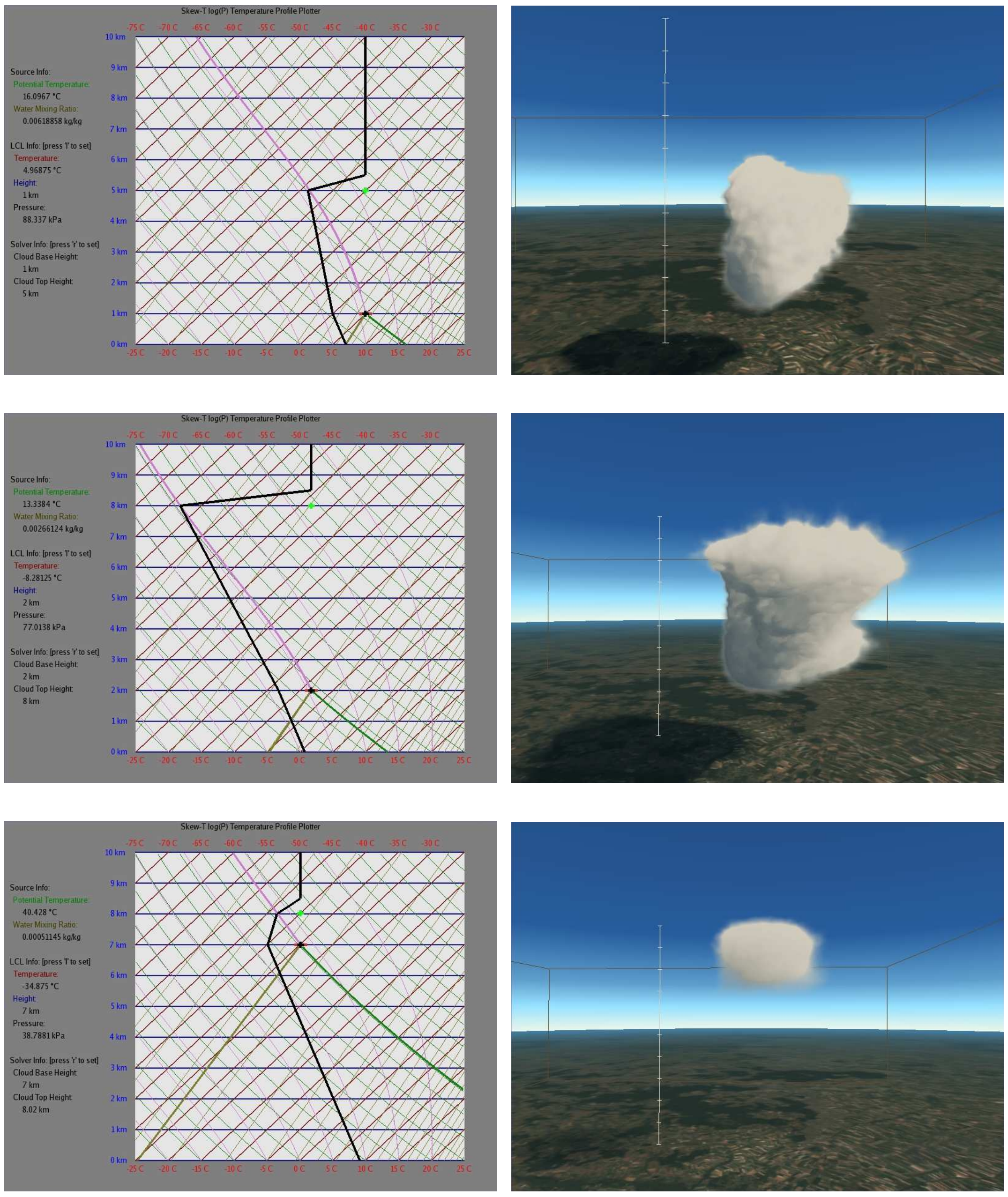
Figure 4.22: Top left, photograph of a large cumuliform cloud from above. Note the texture of the regions with high frequency detail compared with other smoother parts. Top right, simulation. The multiple water categories allow us to use different rendering techniques, to emulate highly textured cloud water and smooth snow regions. Lower — photograph on left, simulation on right, both images have been gamma-adjusted to match. Notice the crisp new cumuliform growth at the bottom of the tall cloud mass, compared with the soft and smooth cloud top. These same features are visible in the simulation, where the crisp features are cloud water and the smoother features are snow. Photographs by the author, July 2009 and April 2012.

4.5.3 Temperature Profile Tool

Sample results of the temperature profile tool are shown in Figure 4.23, which shows that the tool is able to control the vertical extent of the cloudy region.

As mentioned earlier in Section 4.4.2, the user could also 'paint' where the source temperature and vapour should be seeded. This gives a good level of control over where clouds form and by adjusting these values it is also possible to influence the vertical extent of the cloudy region, as demonstrated in Figure 4.24. Decreasing the source temperature while keeping source water constant results in a decrease in vertical extent and base height. Similarly, decreasing the source water level while keeping the temperature constant results in an *increase* in base height and a decrease in vertical extent. These results are as expected from examining a Skew-T chart.

The Skew-T tool can also be used to draw arbitrary temperature profiles. Once the source conditions are set, the LCL is defined and the moist adiabat from this point is highlighted. Portions of the temperature profile above the LCL and to the left of the moist adiabat are where cloud will form, while regions where the profile approaches or crosses this curve will be largely cloud-free. This can be used to achieve layering effects, as demonstrated in Figure 4.25.



Source Info:
Potential Temperature:
16.0967 °C
Water Mixing Ratio:
0.00618858 kg/kg

LCL Info: [press 'l' to set]
Temperature:
4.96875 °C
Height:
1 km
Pressure:
88.337 kPa

Solver Info: [press 'r' to set]
Cloud Base Height:
1 km
Cloud Top Height:
5 km

Source Info:
Potential Temperature:
13.3384 °C
Water Mixing Ratio:
0.00266124 kg/kg

LCL Info: [press 'l' to set]
Temperature:
-8.28125 °C
Height:
2 km
Pressure:
77.0138 kPa

Solver Info: [press 'r' to set]
Cloud Base Height:
2 km
Cloud Top Height:
8 km

Source Info:
Potential Temperature:
40.428 °C
Water Mixing Ratio:
0.00051145 kg/kg

LCL Info: [press 'l' to set]
Temperature:
-34.875 °C
Height:
7 km
Pressure:
38.7881 kPa

Solver Info: [press 'r' to set]
Cloud Base Height:
7 km
Cloud Top Height:
8.02 km

Figure 4.23: Using the reverse Skew-T tool to adjust the temperature profile, Left, to give a cloud formation, Right. The maximum vertical extent is 10km, with increments on the white vertical line in the right-hand diagrams at 1km. In the top pair, the cloud base is set at 1km and the top at 5km. In the middle pair, the base is set to 2km and the top is set to 8km. In the lower pair, the base is set to 7km and the top to 8km. Note the level of overshoot at the top boundary, including the last example where the cloud rises up to meet the top of the simulation domain. This is a limitation of the system.

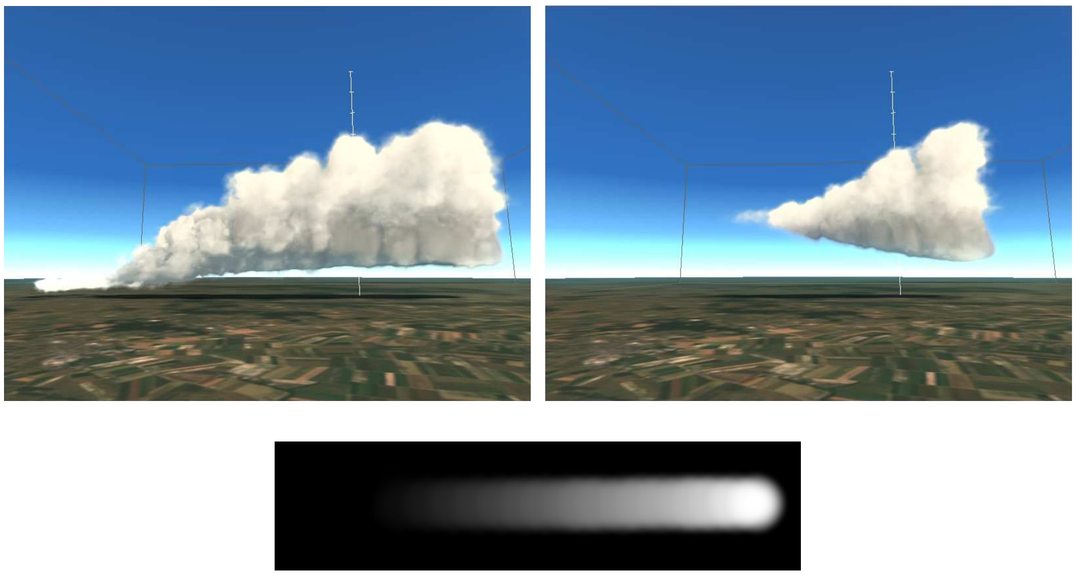


Figure 4.24: Varying the source temperature or the source vapour, ‘painted’ on the ground plane as in the lower image with the simulation results after approximately 2mins shown above. For the painted values, black is equivalent to the background atmospheric temperature and pure white is the ‘seed’ temperature.

4.6 Evaluation: A Critical Analysis

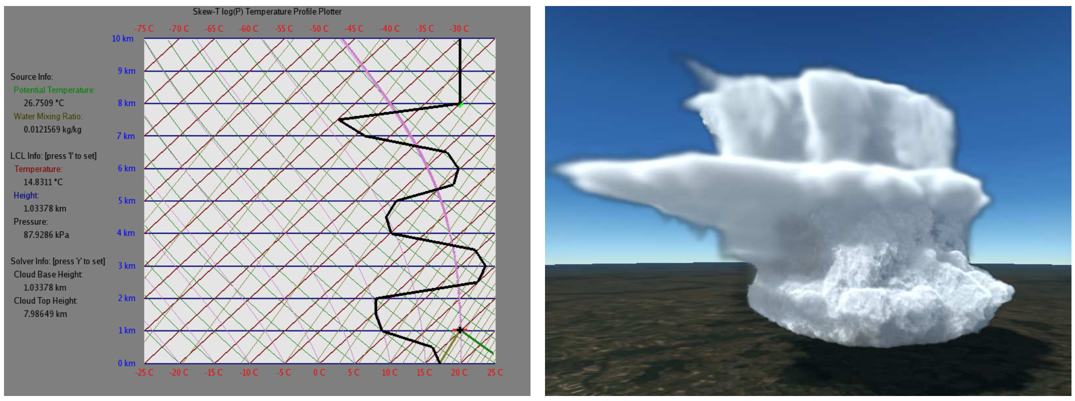
As shown by the results, the methods presented for simulating clouds allow new types and forms to be emulated, in comparison to existing works within computer graphics, and for some level of artistic control to be achieved in directing the resulting formation.

4.6.1 Water Model

The water model presented here is an extension of existing works, which only allowed water in the forms of vapour and cloud water. In these works, this restricted the cloud formations to only those that are non-precipitating. Additionally, high-level ice clouds could not be simulated correctly, because ice crystals take longer to sublimate than water droplets take to evaporate, which should result in longer-lasting clouds with a wispy appearance. In comparison, the six-state water model presented in this work can emulate all of these, adding the possibility of storm cloud formation for the first time. The results show that the new water model is effective at emulating the appearance of a range of cloud forms, including those with precipitation.

One limitation of the model comes from the assumptions of the buoyancy equation. Here, it is assumed that the ambient environment temperature is that of the background atmospheric profile. For a region surrounded by other warm air, this will not be the case and the equation will therefore artificially inflate the buoyancy term. The result of this is that large clouds will essentially become larger, certainly in height. A more correct treatment would evaluate the temperature of neighbouring cells.

As previously described in Section 4.4.2 and shown in Figure 4.24 on the previous page, the prototype tool allows the user to 'paint' where the source temperature and vapour should be seeded. This gives a good level of control over where clouds form and by adjusting these values it is also possible to influence the vertical extent of the cloudy region, as demonstrated in Figure 4.24. Decreasing the source temperature while keeping source water constant results in a decrease in vertical extent and base height. Similarly, decreasing the source water level while keeping the temperature constant results in an *increase* in base height and a decrease in vertical extent. These results are as expected from examining a Skew-T chart.



Source Info:
Potential Temperature:
26.7509 °C
Water Mixing Ratio:
0.0121569 kg/kg

LCL Info: [press 'l' to set]
Temperature:
14.8311 °C
Height:
1.03378 km
Pressure:
87.9286 kPa

Solver Info: [press 'r' to set]
Cloud Base Height:
1.03378 km
Cloud Top Height:
7.98649 km

Figure 4.25: Layering effects may be achieved by manually adjusting the temperature profile. Cloud formation is enhanced in regions where the profile is to the left of the moist adiabat rising from the LCL (the thick pink line in the Skew-T view) and it is reduced in regions to the right of the curve. This example gives three layered regions, although the mid-height one is the best defined.

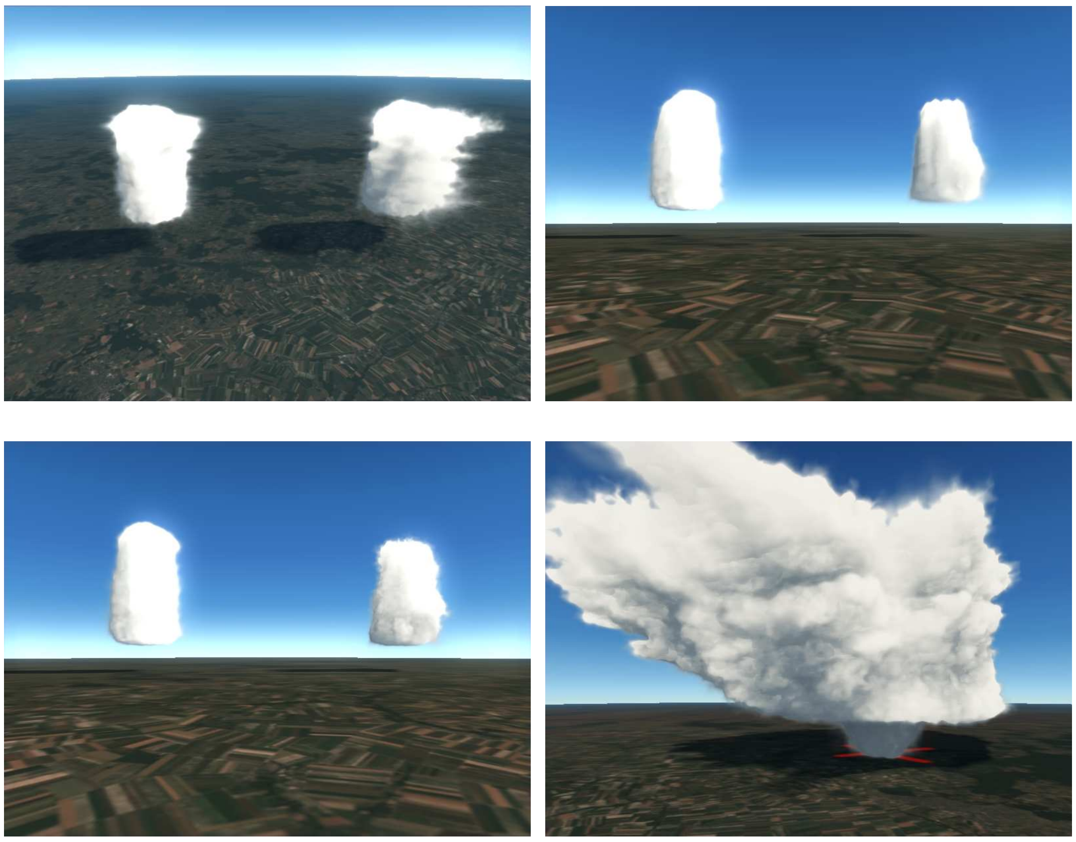


Figure 4.26: Results of adjusting the cloud water control parameters are generally subtle, though the precipitation parameters are much more effective. Top-Left, all parameters enabled in the left cloud and vapour parameter disabled in the right cloud; without evaporation the cloud becomes larger and more wispy at the edges. Top-Right, all parameters enabled in the left cloud and cloud water disabled in the right cloud; consisting only of ice, the cloud base starts higher when the temperature reaches freezing. Lower-Left, all parameters enabled in the left cloud and cloud ice disabled in the right cloud; with no ice formation the cloud does not expand as much above the freezing height. Lower-Right, the results of forcing precipitation to fall on a specific location; here the precipitation parameters are all set to zero except for above the red ‘X’ on the ground plane.

Visual results from adjusting the water control parameters are shown in Figure 4.26, which also indicates the level of control achieved. Adjusting the cloud and vapour parameters produces very subtle effects, while the effect of adjusting the precipitation parameters is much more obvious.

To understand the implications of adjusting the parameters, it is useful to re-examine the transfer diagram in Figure 4.10. The effects of the parameters are as follows:

- Setting the vapour parameter to zero means that clouds essentially do not dissipate fully, although they can still reduce in size through water loss due to precipitation.
- Setting the cloud liquid parameter to zero will completely disable cloud formation in warm conditions and means that a cloud will only form at an altitude when the temperature drops below freezing. This means the cloud base will be different to that specified using the skew-T tool. Re-examination of the continuity equations reveals that disabling cloud liquid also inhibits much of the formation of hail and rain.
- Disabling cloud ice has a similar effect to that of cloud water, but disables cloud formation in cold conditions. For clouds with their base below the freezing point, this may not be readily visible, since cloud liquid can still rise beyond the freezing point even though its transfer into cloud ice is inhibited. Looking back at the continuity equations shows that disabling cloud ice also completely disables snow, which reduces hail and rain formation.
- Setting each of the precipitation parameters to zero naturally disables that category, although again because they are all interdependent disabling one will impact the others. It may help to consider as a general 'rule of thumb' three precipitation life-cycles starting at three generalised altitudes: starting at low level, we simply have rain; starting at mid-altitude we have hail which melts into rain as it falls; finally at high-altitude we start with snow, which converts to hail at mid-altitude which converts to rain at low-altitude. Disabling one or more precipitation states then impacts the life-cycles. It is also worth noting that clouds will generally be much larger when all precipitation categories are disabled, simply due to the retention of water that would otherwise be removed.

One of the most appealing uses of adjusting the water parameters is the ability to force precipitation on a specific location, as shown in the final image in Figure 4.26. It is tempting to suggest this could be used to continuously rain on a moving character for comic effect, however the prototype tool was not designed to support animated parameters so this could not be tested. To do this effectively a number of additional issues would have to be overcome. For instance, the scale of the simulation means it takes some time for rain to fall. If the cloud itself is always above the character, this means that the rain would constantly fall behind it as it moves. Without changing the simulation scale or fall-rate, a solution to this could be to force rain just in front of the character, so the cloud appears to anticipate the character's movement. A more complete investigation of this is left for future work.

The main limitation of the water model controls is that the results are sometimes subtle and difficult to predict, as is the nature of a physical simulation. For example, while it is certainly possible to force rain upon a location (as shown in the results), a cloud of sufficient size must first be built up over the location. To compound the issue, a cloud is generally reluctant to precipitate over its own source, due to the powerful updraught that occurs there. This means that a wind force must be added, to allow the precipitation to fall outside the updraught. The cloud's vapour and heat source must then be upwind from the desired rain target area, making their placement a rather hit-and-miss affair. The control parameters and capabilities are therefore present, but the currently limited tools do not make the task an easy one. Solutions to this problem are left for future work.

4.6.2 Temperature Profiles

Previous works in computer graphics have used a fixed lapse rate for the atmospheric temperature and pressure profiles. This made it impossible to control the vertical extent of the cloudy region. This work presented the use of a user defined temperature profile, to give a level of vertical control over the cloud formation. The results have shown that, for the most part, this is an effective method of achieving this goal.

While the cloud base is clearly defined by the tool the ascending cloud can experience a very large overshoot above its top level of equilibrium, before it falls down to reach it. This is simply due to the momentum gained during its ascent and is a phenomenon observed in nature. This can cause problems, however, if the base of the cloudy region is specified to start at a high altitude, because the momentum gained during the ascent results in a very high overshoot which frequently encounters the top of the simulation domain and results in artefacts (at best a flat-top, although a cloud can become 'stuck' to the top depending on the boundary conditions). The simplest solution to this problem is to extend the vertical limits of the simulation domain. The immediate implication of this is an increase in memory and computational requirements. If this is not an option, the temperature profile could be adjusted to ensure that the temperature difference between the rising parcel and the atmosphere (and thus the buoyancy force) remains low throughout the rising region, in an effort to reduce the momentum. Another solution may be the inclusion of an additional source layer at altitude so that the fluid need not rise as far. Physically, this could be considered to mimic the effects of solar heating and would allow for the proper formation of cirriform clouds.

The fact that the control is only vertical gives the limitation that two vapour and heat sources of equal size will result in clouds of similar vertical extent. If a scene with clouds of varying sizes is required, the only way of achieving this is by modifying the source vapour and heat conditions. In order to determine the resulting vertical extent, careful examination of the Skew-T is required. To achieve specific results, this method is difficult at best. The level of control provided therefore remains a ‘guidance’ rather than a strict regimented adherence.

An assumption of the prototype tool is that changing the temperature profile requires a reset of the simulation. This reset was forced due to initial experiments in which the simulation became unstable when the profile was changed part-way through a simulation run. It would be interesting to properly investigate this effect, to see if a gradual adjustment could maintain simulation stability. The result would be a more varied cloudscape and the possibility to emulate larger-scale weather fronts. Such investigations are left for future work.

The adoption of a user defined temperature profile is a significant step for directing clouds. The Skew-T tool itself is an extremely useful step up from simply editing numbers in a text file, but it is felt that it falls short of being an effective artist’s tool by itself. The graph is still too far removed from the results, making it difficult for an untrained user to imagine what type of cloud formation their profile will result in. The amount of information presented to the user also gives it a rather daunting appearance. In conclusion, the tool is effective in the level of control it presents, but more work is required to form it into an efficient tool for use by an artist.

Chapter 5

Conclusion

In this work we have looked at simulating cloud dynamics, complete with precipitation. We started with a taxonomy of clouds, where we briefly looked at the main cloud types. In Chapter 2, we investigated meteorological approaches to cloud formation and combined these to build a parcel model to mathematically describe our cloud formation process. To do this, we started with a description of dry atmospheric conditions, which revealed how parcels travel through the atmosphere. We examined how the atmospheric temperature profile determines regions of stability, which affect the buoyancy of our parcels. We then added water to our parcels and considered the effects of moist air. We looked at the various forms that water can take and how water changes between these forms and states according to a water continuity model. Armed with our parcel model, we then investigated the general formation and development of cumuliform clouds, the lifecycle of a stormcell and then stratiform and cirriform formations.

In Chapter 3, we reviewed previous works on relevant topics within computer graphics, in particular focusing on cloud simulation, weather effects and fluid dynamics.

Finally, in Chapter 4 we constructed a new simulation scheme, using the parcel model we built up in Chapter 2. This also used a water continuity model, adapted from a meteorological source, which could also be used to provide a level of control to the simulation. A tool was also developed to help control the cloud base and height, by directly manipulating the atmospheric temperature profile. Results were presented, together with photographs of real clouds for comparison, and the effectiveness of the techniques was discussed.

The purpose of this work was to provide a more complete cloud and weather model, by developing and adapting techniques that would allow the emulation of a wider variety of atmospheric phenomena than was previously possible. To achieve this, a physically-based model was developed, taking the lead from earlier works. However, computer graphics for entertainment is not always concerned with pure physical simulation. We strive to produce visually plausible results and do not need the numerical accuracy that a detailed scientific model offers. What we do need is the ability to direct our outputs to achieve artistic goals and sometimes these goals are contrary to our reality. In most cases, the reality we wish to produce is not far from our own, so a simple bending of the laws of physics is sufficient. Thus, a physically-based system was used as a starting point and appropriate methods of control were also investigated.

5.0.3 Contributions

In summary, the main contributions of this work are:

- Water model that uses multiple water categories:
  - Provides six states of water: vapour, cloud water, cloud ice, rain, snow and hail.
  - A lightweight bulk-water continuity model, built on physically-based meteorological work. This is a system of equations that govern how water is transferred from one category to another.
  - The water model allows us to emulate not just cloud formations but also connected weather phenomena such as storm clouds that actually precipitate.
  - Individual clouds in this work can consist of multiple water types, which may be visualised differently and can behave differently. For example, the precipitates can fall at different rates, so rain and hail quickly reach the ground while snow lingers at high levels.
  - By explicitly and separately simulating these water categories, it is possible to use multi-field rendering techniques to visualise a whole range of atmospheric phenomena. This means that with this work a single dynamic model can now show features such as rainbows, ice haloes, glories, as well as the direct visuals of precipitation. Previously, this was only possible by either rendering the direct outputs from scientific simulations, or through modelling techniques where clouds are explicitly defined.
- Artistic control within the water model:
  - Controls placed within the water model influence the flow of water between states.
  - This allows the user to influence the type of cloud that would form.

  - It also allows the user to indicate where they want rain to fall, for example, providing they have also set up a suitable cloud. This was not possible previously, because models did not provide precipitation water categories or any means of influencing the flow between categories.

- Height control and tool:
  - A user defined temperature profile is used to describe the background atmospheric temperature.
  - This provides vertical regions of stability and instability which directly affect the vertical extent of cloud formations.
  - This gives a degree of control over the resulting simulation, allowing for more varied and visually interesting cloud formations.
  - A simple interactive tool was developed that can be used to control the base and height of a cloud formation. This was not possible previously, because models used a single lapse rate to describe the vertical temperature profile.

5.1 Discussion, Limitations and Further Work

A number of points have arisen from the work undertaken here and there are many directions for future work that can be considered.

One obvious visual artefact, which cannot be seen in still images, are the 'jumps' caused by the decoupling of the simulation update rate from the rendering process. The original intention was that these would be linearly interpolated from one simulation frame to the next within the rendering technique. However, the prototype was typically run at ten to thirty times faster than real-time and it was felt that pure linear interpolation would be unable to correctly blend from one volume to the next when the cloud properties move further than one cell-width at a time. Harris [2003] identified this problem and also suggested using the advection output from the previous fluid simulation step to further advect the cloud properties as an approximation between actual simulation updates. It is possible that this would give reasonable results but, as Harris noted, this would also be more computationally involved than simple linear interpolation. This feature has therefore been left unimplemented, for consideration as future work.

5.1.1 Issues of Control

This work has attempted to start addressing the problem of how to control a physically-inspired simulation. The very nature of a physically-based simulation is one by which the evolution of the system and its outputs are entirely controlled by the laws of ‘nature’ that are programmed into the model. This is fundamentally at odds with the requirements of production computer graphics. Here, it is not just desirable but absolutely necessary that such simulations can be directed, or artistically controlled, in order to achieve specific visual goals.

As shown in the results and evaluation, the level of control that this work offers is a substantial improvement on previous cloud works but the control is still rudimentary. Note that the level of control offered by Dobashi et al. [2008] is quite absolute. Their clouds will grow to conform to the user-drawn contour. While this level of control can be highly desirable, we strongly argue that not every cloud needs to be directed so rigidly. It would be very artist-intensive to produce a naturalistic and dynamic cloudscape, when in reality only a few individual clouds actually need this level of control. Such control also completely prohibits ‘on the fly’ cloud generation, which could be useful in flight simulators and other real-time environments.

An ideal system would be one where different levels of control are available. It might be that sometimes artists need to directly manipulate the shape of clouds, or to force precipitation, while at other times a more gentle direction is all that is required. These could both be needed within the same scene — for example so that ‘hero clouds’ in the foreground are crafted into just the right shape, while more distant clouds in the background are pure physically-based outputs.

Determinism is also an interesting control consideration, though perhaps more for offline applications where it could be useful when synchronising between shots or if a simulation must be re-run at a later date. The prototype simulation developed here is non-deterministic, two runs of the same settings producing similar but not identical cloud formations. The importance and application of this issue remains open for future work.

5.1.2 Wind

The work presented here has only made light attempts at addressing issues of wind, which could potentially be a good source of artistic control. As mentioned briefly in Chapter 3, attempts have already been made at directing fluid motions so it would be interesting to expand this into the subject of cloud formations. Of particular interest would be to set up a large rotating air mass, to emulate a supercell and to investigate extreme weather such as tornadoes.

The concept of ‘wind’ that has been used in this work has been a general attempt at emulating air movements that are much larger than the simulation domain. Weather fronts, caused by large-scale differences in temperature and pressure, can cause a variety of interesting cloud effects and so it would be interesting to investigate some of these. This would likely involve two different temperature profiles and some form of boundary to specify the different regions.

In general, it is relatively easy for a single cloud to be blown from one side of the simulation domain to another. As a ground-based observer, it is easy to see that clouds move from one crowded horizon to another. A proper treatment of wind would need to consider this, by either expanding the simulation domain or devising some scheme to allow fully formed clouds to enter the domain.

5.1.3 Simulation Domain

Simulation domain size is a key issue with the Eulerian technique. From simple geometry and without taking into account effects such as atmospheric refraction, we know that from a point on the surface of the Earth you can see the top of the troposphere (assuming this is 10km high) to a distance of around 350km. Thus, if we stand on the ground and see a very high-level cirrus cloud exactly on the horizon, it could theoretically be about 350km away. Practically, scattering from haze and other atmospheric particles (including water) will limit the actual visibility, but the issue remains that this is still a sizeable domain. If we ensure the viewer always sees a 'cluttered horizon', we could greatly limit this, but the problem is greatly exacerbated if the viewer is above ground such as in an aeroplane.

The difficulties, of course, are the memory and computational requirements of simulating such a large volume at a resolution sufficient to produce clouds of any detail. The maximum resolution this work was run at was 240x240x80 = 4.6million voxels, over a 30x30x10km domain. If we assume just a 500km square domain at the same resolution, 1.3billion voxels would be required. Based on the memory usage of the prototype, this resolution might require around 250GB of graphics memory. The prototype was not designed to be particularly efficient in its use of resources, but this scale is out of reach for off-the-shelf hardware at the time of writing and for the foreseeable future. The problem then becomes one of how to reduce this requirement.

Non-regular grids could be worth investigating, in a level-of-detail type approach. Here, the voxel size in the distance could be much larger than in the centre, effectively putting the resolution where it is needed: at the point where the viewer sees it up close. The use of irregularly-sized voxels would impact the fluid solver, though such techniques have been used in computational fluid dynamics for some time. The more relevant impact would be to the cloud model, which would need to ensure that water mass is correctly maintained during advection, without great changes in density when moving from voxels of different size, and possibly the use of voxel size in the state transfer equations. This would need careful investigation to ensure a consistent behaviour and appearance across the entire simulation domain. Again, the lead could be taken from meteorology, where non-regular grids have been used before [Xue et al. 2000]. The obvious disadvantage with such techniques is the added computational expense, though how much this would be is unclear at this point, but the saving in memory could be useful.

It should also be possible to 'force perspective', by reducing the actual height of the distant tropopause to make the more distant clouds appear to be further away. This effectively makes the radius of the Earth smaller, but it is possible that the effect could be subtle enough that most viewers would not notice and this could therefore be a useful resource-saver if it were executed with skill.

5.1.4 Procedural Detail

Higher resolution grids allow more detail to be simulated, and thus visualised, but given the domain requirements there will clearly always be a shortfall in the amount of resolution that is possible. One alternative solution is the use of high-frequency procedural detail, some techniques for which have been discussed in Chapter 3.

A key concept that emerged from this work was the question of whether it would be possible to link the controls of the procedural detail into the simulation. Different visual results could then be obtained for different water categories. Procedural streaks could be generated for thin cloud ice, to indicate Cirrus formations, possibly with the texture coordinates advected by the fluid simulation to correctly emulate the streak direction.

A more involved procedural problem would be to correctly add detail to cloud water and cloud ice in high densities, for deeply convecting cumuliform clouds. The aim here would be to provide the classic 'cauliflower top' as the new cloud expands and there is good contrast between the high water density of the cloud and the surrounding atmosphere. As the cloud ages, the edges need to become less well-defined and the detail should move away from the cauliflower appearance to become more ragged, as swirls in the fluid atmosphere erode the cloud's edges. The cauliflower style procedural noise should be possible if the cloud 'surface' and normal are known, and could be achieved using distance fields and hypertextures [Bouthors et al. 2008].

If controls can be provided by the simulation system, different procedural techniques may be applied to emulate different high-frequency details. However, many procedural techniques are essentially static, while the detail in real clouds is quite dynamic. While there are procedural techniques that evolve over time [Ebert et al. 2003] a review would need to be undertaken to determine if they are visually 'cloud like' and any set of techniques used would also need to work together to provide a seamless appearance.

References

Acheson, D. 1990. *Elementary Fluid Dynamics*. Oxford Applied Mathematics and Computing Science Series. Oxford University Press, Oxford, UK.

Air Weather Service. 1990. The Use of The Skew T, Log P Diagram in Analysis and Forecasting. United States Air Force, Scott AFB.

AMD-ATI. 2002. Clouds - Technical Demonstration. Available from: `http://ati.amd.com/developer/samples/clouds.html` [Accessed 16 Dec 2008].

Anderson, E. F. and McLoughlin, L. 2006a. C-Sheep: Controlling Entities in a 3D Virtual World as a Tool for Computer Science Education. In *Proceedings of Future Play 2006 posters*.

Anderson, E. F. and McLoughlin, L. 2006b. Do Robots Dream of Virtual Sheep: Rediscovering the "Karel the Robot" Paradigm for the "Plug&Play Generation". In *Proceedings of the Fourth International Game Design and Technology Workshop and Conference (GDTW2006)*.

Anderson, E. F. and McLoughlin, L. 2007. Critters in the Classroom: a 3D Computer-game-like Tool for Teaching Programming to Computer Animation Students. In *SIGGRAPH '07: ACM SIGGRAPH 2007 educators program*, ACM, New York, NY, USA, p. 7.

Andrews, D. G. 2000. *An Introduction to Atmospheric Physics*. Cambridge University Press, Cambridge, UK.

Angelidis, A., Neyret, F., Singh, K. and Nowrouzezahrai, D. 2006. A Controllable, Fast and Stable Basis for Vortex Based Smoke Simulation. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, pp. 25–32.

Auld, D. and Srinivas, K. ca.2008. Properties of the Atmosphere. Available from: `http://www-mdp.eng.cam.ac.uk/web/library/enginfo/aerothermal_dvd_only/aero/atmos/atmos.html` [Accessed 15 Dec 2008].

Bell, G. 1998. Creating Backgrounds for 3D Games. Available from: `http://www.gamasutra.com/features/19981023/bell_01.htm` [Accessed 16 Dec 2008].

Benenson, W., Harris, J. W., Stocker, H. and Holger, L. 2006. *Handbook of Physics*. Springer-Verlag, New York, NY, USA.

Blinn, J. F. 1982. Light Reflection Functions for Simulation of Clouds and Dusty Surfaces. *SIGGRAPH Comput. Graph. 16*(3), pp. 21–29.

Bohren, C. F. and Albrecht, B. A. 1998. *Atmospheric Thermodynamics.* Oxford University Press, New York, NY, USA.

Bolton, D. 1980. The Computation of Equivalent Potential Temperature. *Monthly Weather Review 108*(7), pp. 1046–1053.

Bouthors, A., Neyret, F. and Lefebvre, S. 2006. Real-time realistic illumination and shading of stratiform clouds. In *Eurographics Workshop on Natural Phenomena.*

Bouthors, A., Neyret, F., Max, N., Bruneton, E. and Crassin, C. 2008. Interactive multiple anisotropic scattering in clouds. In *ACM Symposium on Interactive 3D Graphics and Games (I3D).*

Brickman, N., Olsen, D. and Smith, G. 2007a. Realtime Cloud Simulation and Rendering. Available from: `http://www.soe.ucsc.edu/classes/cmps262/Winter07/reviews/clouddraftpaper.pdf` [Accessed 17 Dec 2008].

Brickman, N., Olsen, D. and Smith, G. 2007b. Shapes in the Clouds: Interactive, Artist-Guided Cloud Simulation. Available from: `www.soe.ucsc.edu/classes/cmps262/Winter07/projects/clouds/finaldraft.pdf` [Accessed 17 Dec 2008].

Bridson, R. 2008. *Fluid Simulation for Computer Graphics.* AK Peters, Wellesley, MA, USA.

Buck, A. L. 1981. New Equations for Computing Vapor Pressure and Enhancement Factor. *Journal of Applied Meteorology 20*(12), pp. 1527–1532.

Byers, H. R. and Braham, Jr, R. R. 1948. Thunderstorm Structure and Circulation. *Journal of the Atmospheric Sciences 5*(3) (June), pp. 71–86.

Co, C. S., Hamann, B. and Joy, K. I. 2003. Iso-Splatting: A Point-based Alternative to Isosurface Visualization. In *In Proceedings of the Eleventh Pacific Conference on Computer Graphics and Applications - Pacific Graphics 2003*, pp. 325–334.

Cowley, L. ca.2008. Atmospheric Optics. Available from: `http://www.atoptics.co.uk/` [Accessed 15 Dec 2008].

Cuntz, N., Kolb, A., Leidl, M., Rezk-Salama, C. and Böttinger, M. 2007. GPU-based Dynamic Flow Visualization for Climate Research Applications. In *SimVis*, SCS Publishing House e.V., T. Schulze, B. Preim, and H. Schumann, Eds., pp. 371–384.

Desbrun, M. and Cani, M.-P. 1996. Smoothed Particles: a new Paradigm for Animating Highly Deformable Bodies. In *Proceedings of the Eurographics workshop on Computer animation and simulation '96*, Springer-Verlag New York, Inc., New York, NY, USA, pp. 61–76. Published under the name Marie-Paule Gascuel.

Digital Dutch. ca.2008. US Standard Atmosphere Calculator. Available from: `http://www.digitaldutch.com/atmoscalc/` [Accessed 15 Dec 2008].

Dobashi, Y., Nishita, T. and Okita, T. 1998a. Animaiton of Clouds Using Cellular Automaton. In *Computer Graphics and Imaging*, pp. 25–256.

Dobashi, Y., Nishita, T., Yamashita, H. and Okita, T. 1998b. Modeling of Clouds from Satellite Images using Metaballs. In *PG '98: Proceedings of the 6th Pacific Conference on Computer Graphics and Applications*, IEEE Computer Society, Washington, DC, USA, p. 53.

Dobashi, Y., Kaneda, K., Yamashita, H., Okita, T. and Nishita, T. 2000. A simple, efficient method for realistic animation of clouds. In *ACM SIGGRAPH 2000: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, pp. 19–28.

Dobashi, Y., Yamamoto, T. and Nishita, T. 2001. Efficient Rendering of Lightning Taking into Account Scattering Effects due to Cloud and Atmospheric Particles. In *PG '01: Proceedings of the 9th Pacific Conference on Computer Graphics and Applications*, IEEE Computer Society, Washington, DC, USA, p. 390.

Dobashi, Y., Kusumoto, K., Nishita, T. and Yamamoto, T. 2008. Feedback control of cumuliform cloud formation based on computational fluid dynamics. In *ACM SIGGRAPH 2008*, ACM, New York, NY, USA, pp. 1–8.

Ebert, D. S., Musgrave, F. K., Peachey, D., Perlin, K. and Worley, S. 2003. *Texturing and Modelling: A Procedural Approach*, third ed. The Morgan Kaufmann Series in Computer Graphics and Geometric Modelling. Morgan Kaufmann, San Francisco, CA, USA.

ECMWF. ca.2013. Research at the European Centre for Medium-range Weather Forecasts. Available from: `http://www.ecmwf.int/research/` [Accessed 15 Jun 2013].

Elias, H. 1998. Cloud Cover. Available from: `http://freespace.virgin.net/hugo.elias/models/m_clouds.htm` [Accessed 16 Dec 2008].

Elinas, P. and Stürzlinger, W. 2000. Real-time rendering of 3D clouds. *J. Graph. Tools 5*(4), pp. 33–45.

Emanuel, K. A. 1994. *Atmospheric Convection*. Oxford University Press, New York, NY, USA.

Engel, K., Hadwiger, M., Kniss, J. M., Rezk-Salama, C. and Weiskopf, D. 2006. *Real-Time Volume Graphics*. AK Peters, Wellesley, MA, USA.

Fearing, P. 2000. Computer modelling of fallen snow. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, pp. 37–46.

Fedkiw, R., Stam, J. and Jensen, H. W. 2001. Visual simulation of smoke. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, pp. 15–22.

Fernando, R. and Kilgard, M. J. 2003. *The Cg Tutorial*. Addison Wesley, Boston, MA, USA.

Foster, N. and Metaxas, D. 1997. Modeling the Motion of a Hot, Turbulent Gas. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, pp. 181–188.

Frei, S. ca.2008. Swiss Aviation Resources ICAO Standard Atmosphere Calculator. Available from `http://www.aviation.ch/tools-atmosphere.asp` [Accessed 15 Dec 2008].

Gardner, G. Y. 1985. Visual Simulation of Clouds. *SIGGRAPH Comput. Graph. 19*(3), pp. 297–304.

Garg, K. and Nayar, S. 2006. Photorealistic Rendering of Rain Streaks. *ACM Trans. on Graphics (also Proc. of ACM SIGGRAPH)* (Jul).

Gingold, R. and Monaghan, J. 1977. Smoothed Particle Hydrodynamics: Theory and Application to Non-Spherical Stars. *Monthly Notices of the Royal Astronomical Society 181*, pp. 375–389.

Goodnight, N. 2007. CUDA/OpenGL Fluid Simulation. NVIDIA CUDA SDK v1.0 Technical Demonstration Whitepaper. Available from: `http://docs.nvidia.com/cuda/samples/5_Simulations/fluidsGL/doc/fluidsGL.pdf` [Accessed 15 Jun 2013].

Griebel, M., Dornseifer, T. and Neunhoeffer, T. 1997. *Numerical Simulation in Fluid Dynamics, A Practical Introduction.* Series on Mathematical Modelling and Computation. SIAM: Society for Industrial and Applied Mathematics, Piladelphia, USA.

Hamblyn, R. 2002. *The Invention of Clouds.* Picador, London, UK.

Harada, T., Koshizuka, S. and Kawaguchi, Y. 2007. Smoothed Particle Hydrodynamics on GPUs. In *Proceedings of Computer Graphics International*, pp. 63–70.

Harlow, F. H. and Welch, J. E. 1965. Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface. *Physics of Fluids 8*(12), pp. 2182–2189.

Harris, M. J. and Lastra, A. 2001. Real-Time Cloud Rendering. In *Proceedings of EUROGRAPHICS 2001*, Eurographics Association, vol. 20, pp. 76–84.

Harris, M. J., Coombe, G., Scheuermann, T. and Lastra, A. 2002. Physically-based visual simulation on graphics hardware. In *HWWS '02: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, pp. 109–118.

Harris, M. J., Baxter, W. V., Scheuermann, T. and Lastra, A. 2003. Simulation of cloud dynamics on graphics hardware. In *HWWS '03: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, pp. 92–101.

Harris, M. J. 2002. Real-Time Cloud Rendering for Games. In *Proceedings of Game Developers Conference 2002.*

Harris, M. J. 2003. *Real-Time Cloud Simulation and Rendering.* PhD thesis, University of North Carolina at Chapel Hill, NC, USA.

Harris, M. J. 2004. Chapter 38: Fast Fluid Dynamics Simulation on the GPU. In *GPU Gems: Programming Techniques, Tips and Tricks for Real-Time Graphics*, R. Fernando, Ed. Addison-Wesley Professional, pp. 637–665.

Heinzlreiter, P., Kurka, G. and Volkert, J. 2002. Real-time Visualization of Clouds. In *WSCG (Short Papers)*, pp. 43–50.

Hollingshead, M. ca.2009. Extreme Instability (storm-chasing gallery). Available from: `http://www.extremeinstability.com` [Accessed 28 Jan 2009].

Horvath, P. and Illes, D. 2007. SPH-Based Fluid Simulation for Special Effects. In *CESCG 2007: The 11th Central European Seminar on Computer Graphics*. Available from: `http://www.cg.tuwien.ac.at/hostings/cescg/CESCG-2007/papers/TUBudapest-Horvath-Peter/TUBudapest-Horvath-Peter.pdf` [Accessed 17 Dec 2008].

Houze, Jr., R. A. 1993. *Cloud Dynamics*. International Geophysics Series. Academic Press, San Diego, CA, USA.

ICAO. ca.2008. ICAO Standard Atmosphere. Available from: `http://www.icao.int/icao/en/cd_pub_list.htm` [Accessed 15 Dec 2008].

ISO. ca.2008. ISO 2533:1975 Standard Atmosphere. Available from: `http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=7472` [Accessed 15 Dec 2008].

Kajiya, J. T. and Herzen, B. P. V. 1984. Ray tracing volume densities. In *ACM SIGGRAPH 84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, pp. 165–174.

Kaplan, K. G. 1943. On the Mechanisms of Thunderstorm Downdrafts. *Bulletin of the American Meteorology Society 24*, pp. 54–59.

Kass, M. and Miller, G. 1990. Rapid, Stable Fluid Dynamics for Computer Graphics. In *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, pp. 49–57.

Kersthold, J. 2004. Photograph: Rolling Thunderstorm. Available from: `http://upload.wikimedia.org/wikipedia/commons/d/da/Rolling-thunder-cloud.jpg` [Accessed 15 Dec 2008].

Kim, T. and Lin, M. C. 2004. Physically Based Animation and Rendering of Lightning. In *PG '04: Proceedings of the Computer Graphics and Applications, 12th Pacific Conference*, IEEE Computer Society, Washington, DC, USA, pp. 267–275.

Kim, Y., Machiraju, R. and Thompson, D. 2006. Path-based Control of Smoke Simulations. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, pp. 33–42.

Kim, T., Thürey, N., James, D. and Gross, M. 2008. Wavelet Turbulence for Fluid Simulation. In *SIGGRAPH '08: ACM SIGGRAPH 2008 papers*, ACM, New York, NY, USA, pp. 1–6.

Kokhanovsky, A. A. 2006. *Cloud Optics*. Springer, Dordrecht, The Netherlands.

Kolb, A. and Cuntz, N. 2005. Dynamic Particle Coupling for GPU-based Fluid Simulation. *Proc. 18th Symposium on Simulation Technique*, pp. 722–727.

Kolb, A., Latta, L. and Rezk-Salama, C. 2004. Hardware-Based Simulation and Collision Detection for Large Particle Systems. In *HWWS '04: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, ACM, New York, NY, USA, pp. 123–131.

Kruger, J. and Westermann, R. 2005. GPU Simulation and Rendering of Volumetric Effects for Computer Games and Virtual Environments. *Computer Graphics Forum 24*(3), pp. 685–693.

Latta, L. 2004. Building a Million Particle System. In *Proceedings of the Game Developers Conference*, pp. 54–60.

Laven, P. ca.2012. Mie Plot. Available from: `http://www.philiplaven.com/mieplot.htm` [Accessed 5 Apr 2012].

Lawrence, M. ca.2008. Atmospheric Thermodynamics and Parcel Theory, Chapter 1. From: Material for the Special Lecture on Deep Convection. Available from: `http://www.mpch-mainz.mpg.de/~lawrence/vorlesung_WS2004-5/thermo.pdf` [Accessed 15 Dec 2008].

Liao, H.-S., Chuang, J.-H. and Lin, C.-C. 2004. Efficient Rendering of Dynamic Clouds. In *VRCAI '04: Proceedings of the 2004 ACM SIGGRAPH international conference on Virtual Reality continuum and its applications in industry*, ACM, New York, NY, USA, pp. 19–25.

Lin, Y.-L., Farley, R. D. and Orville, H. D. 1983. Bulk Parameterization of the Snow Field in a Cloud Model. *Journal of Applied Meteorology 22* (June), pp. 1065–1092.

Losasso, F., Talton, J., Kwatra, N. and Fedkiw, R. 2008. Two-Way Coupled SPH and Particle Level Set Fluid Simulation. *IEEE Transactions on Visualization and Computer Graphics 14*(4), pp. 797–804.

Lucy, L. B. 1977. A Numerical Approach to the Testing of the Fission Hypothesis. *Astronomical Journal 82* (Dec.), pp. 1013–1024.

Magono, C. and Lee, C. W. 1966. Meteorological Classification of Natural Snow Crystals. *Journal of the Faculty of Science, Hokkaido University. Series 7, Geophysics 2*(4), pp. 321–335. Available from: `http://hdl.handle.net/2115/8672` [Accessed 15 Dec 2008].

Manssour, I. H., Freitas, C. M. D. S. and Claudio, D. M. 1996. Tools for Meteorological Data Visualization. In *SIBGRAPI'96 - Proceedings of the Brazilian Symposium on Computer Graphics and Image Processing*.

McLoughlin, L. and Anderson, E. F. 2006. I See Sheep: A Practical Application of Game Rendering Techniques for Computer Science Education. In *Proceedings of Future Play 2006 posters*.

McLoughlin, L. 2008. Cloud 21. In *SIGGRAPH '08: ACM SIGGRAPH 2008 posters*, ACM, New York, NY, USA, pp. 1–1.

McNamara, A., Treuille, A., Popović, Z. and Stam, J. 2004. Fluid Control Using the Adjoint Method. In *ACM SIGGRAPH 2004*, ACM, New York, NY, USA, pp. 449–456.

Met Office. ca.2008. The Atmosphere. Available from: `http://www.metoffice.gov.uk/education/secondary/teachers/atmosphere.html` [Accessed 15 Dec 2008].

Met Office. ca.2013. Met Office Unified Model. Available from: `http://www.metoffice.gov.uk/research/modelling-systems/unified-model` [Accessed 15 Jun 2013].

Meuer, H., Strohmaier, E., Dongarra, J. and Simon, H. 2012. Top 500 Supercomputer Sites. Available from: `http://www.top500.org` [Accessed 15 Jun 2013].

Microsoft. ca.2013. Microsoft Flight Simulator. Available from: `http://www.microsoft.com/games/flight/` [Accessed 10 Jun 2013].

Miller, G. and Pearce, A. 1989. Globular Dynamics: A Connected Particle System for Animating Viscous Fluids. *Computers and Graphics 13*(3), pp. 305–309.

Miyazaki, R., Yoshida, S., Nishita, T. and Dobashi, Y. 2001. A Method for Modeling Clouds Based on Atmospheric Fluid Dynamics. In *PG '01: Proceedings of the 9th Pacific Conference on Computer Graphics and Applications*, IEEE Computer Society, Washington, DC, USA, p. 363.

Miyazaki, R., Dobashi, Y. and Nishita, T. 2002. Simulation of cumuliform clouds based on computational fluid dynamics. In *Proceedings of EUROGRAPHICS 2002 Short Presentations*, pp. 405–410.

Müller, M., Charypar, D. and Gross, M. 2003. Particle-Based Fluid Simulation for Interactive Applications. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, pp. 154–159.

Nagel, K. and Raschke, E. 1992. Self-organizing criticality in cloud formation? In *Physica A: Statistical Mechanics and its Applications*, Elsevier B.V., vol. 182, pp. 519–531.

NASA Landsat Program. 2008. Landsat ETM+ scene p198r026_7dt20010703, L1G, USGS, Sioux Falls. Acquisition Date 2001-07-03. Source for this data set was the Global Land Cover Facility, www.landcover.org.

NASA. ca.2008. US Standard Atmosphere. Available from: `http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19770009539_1977009539.pdf` [Accessed 15 Dec 2008].

National Geographic. 2003. National Geographic News article: Largest Hailstone in U.S. History Found. Available from: `http://news.nationalgeographic.com/news/2003/08/0804_030804_largesthailstone.html` [Accessed 15 Dec 2008].

Neyret, F. 1997. Qualitative Simulation of Convective Cloud Formation and Evolution. In *Proc of Eurographics Computer Animation and Simulation Workshop'97*, pp. 113–124.

Nishita, T., Sirai, T., Tadamura, K. and Nakamae, E. 1993. Display of the earth taking into account atmospheric scattering. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, pp. 175–182.

Nishita, T., Dobashi, Y. and Nakamae, E. 1996. Display of clouds taking into account multiple anisotropic scattering and sky light. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, pp. 379–386.

NOAA. ca.2013. Global Forecasting System. National Climatic Data Centre, The National Oceanic and Atmospheric Administration. Available from: `http://www.ncdc.noaa.gov/data-access/model-data/model-datasets/global-forcast-system-gfs` [Accessed 15 Jun 2013].

NVIDIA. 2007. Metaballs - NVIDIA Direct3D SDK 10 Technical Demonstration. Available from: `http://developer.download.nvidia.com/SDK/10.5/direct3d/samples.html#MetaBalls` [Accessed 16 Dec 2008].

Oleg A. Potiy, A. A. A. 2005. 3D Flow Visualization Using GPU-Driven Particle System. In *GraphiCon: Proceedings of the International Conference on Computer Graphics & Vision 2005*, pp. 25–32.

Olsen, J. C. 2004. Photograph: Mammatus Clouds over Hastings, Nebraska. Available from: `http://www.hprcc.unl.edu/nebraska/june2004hastings-mammatus.html` [Accessed 15 Dec 2008].

Overby, D., Melek, Z. and Keyser, J. 2002a. Interactive physically-based cloud simulation. In *PG '02: Proceedings of the 10th Pacific Conference on Computer Graphics and Applications*, IEEE Computer Society, Washington, DC, USA, p. 469.

Overby, D., Melek, Z. and Keyser, J. 2002b. Interactive Physically-based Cloud Simulation. *Technical Report, Computer Science Department, Texas A & M University*, pp. 2002–7–2.

Overby, D. R. 2002. *Interactive Physically-Based Cloud Simulation.* Master's thesis, Texas A&M University, Texas, USA.

Pagani. 2005. Photograph: Pagani Zonda F - Wind Tunnel. Available from: `http://www.seriouswheels.com/2005/2005-Pagani-Zonda-F-Wind-Tunnel-1600x1200.htm` [Accessed 15 Dec 2008].

Pallister, K. 2001. Generating Procedural Clouds Using 3D Hardware. In *Game Programming Gems 2*, M. DeLoura, Ed. Charles River Media, pp. 463–473.

Papathomas, T. V., Schiavone, J. A. and Julesz, B. 1988. Applications of computer graphics to the visualization of meteorological data. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, pp. 327–334.

Park, S. I. and Kim, M. J. 2005. Vortex Fluid for Gaseous Phenomena. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ACM, New York, NY, USA, pp. 261–270.

Perlin, K. 1985. An image synthesizer. *SIGGRAPH Comput. Graph. 19*(3), pp. 287–296.

Petterssen, S. 1940. *Weather Analysis and Forecasting.* McGraw-Hill Book Co., New York, NY, USA.

Preetham, A. J., Shirley, P. and Smits, B. 1999. A Practical Analytic Model for Daylight. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, pp. 91–100.

Premoze, S., Tasdizen, T., Bigler, J., Lefohn, A. and Whitaker, R. 2003. Particle-Based Simulation of Fluids. *Computer Graphics Forum 22*(3).

Pretor-Pinney, G. 2006. *The Cloudspotter's Guide.* Sceptre, London, UK.

Pretor-Pinney, G. ca.2012. The Cloud Appreciation Society. Available from: `http://cloudappreciationsociety.org` [Accessed 26 April 2012].

Riley, K., Ebert, D., Hansen, C. and Levit, J. 2003. Visually Accurate Multi-Field Weather Visualization. In *VIS '03: Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, IEEE Computer Society, Washington, DC, USA, p. 37.

Riley, K., Ebert, D., Hansen, C. and Levit, J. 2004a. A System for Realistic Weather Rendering. In *Proceedings of the 84th American Meteorological Society Annual Meeting.*

Riley, K., Ebert, D., Kraus, M., Tessendorf, J. and Hansen, C. 2004b. Efficient Rendering of Atmospheric Phenomena. In *Proceedings of Eurographics Symposium on Rendering 2004*, pp. 375–386.

Robertson, B. 2000. Sea Change. *Computer Graphics World 23*(7). Available from: `http://www.cgw.com/Publications/CGW/2000/Volume-23-Issue-7-July-2000-/Sea-Change.aspx` [Accessed 23 Apr 2012].

Roettger, S. and Ertl, T. 2003. Fast volumetric display of natural gaseous phenomena. In *In Computer Graphics International*, IEEE Computer Society, pp. 74–83.

Rogers, R. R. and Yau, M. K. 1989. *A Short Course in Cloud Physics*, third ed. International Series in Natural Philosophy. Butterworth Heinemann, Cambridge, UK.

Schpok, J., Simons, J., Ebert, D. S. and Hansen, C. 2003. A real-time cloud modeling, rendering, and animation system. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, pp. 160–166.

Schultz, P. 1995. An Explicit Cloud Physics Parameterization for Operational Numerical Weather Prediction. *Monthly Weather Review 123*(11), pp. 3331–3343.

Selle, A., Rasmussen, N. and Fedkiw, R. 2005. A Vortex Particle Method for Smoke, Water and Explosions. *ACM Trans. Graph. 24*(3), pp. 910–914.

Simul Software. ca.2008a. CloudWright by Simul Software. Available from: `http://www.simul.co.uk/cloudwright` [Accessed 17 Dec 2008].

Simul Software. ca.2008b. Simul Weather by Simul Software. Available from: `http://www.simul.co.uk/weather` [Accessed 17 Dec 2008].

Srivastava, R. 1971. Size distribution of raindrops generated by their breakup and coalescence. *Journal of the Atmospheric Sciences 28*(3), pp. 410–415.

Stam, J. 1991. *A Multi-Scale Stochastic Model for Computer Graphics*. Master's thesis, University of Toronto, Toronto, ON, Canada.

Stam, J. 1999. Stable fluids. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, pp. 121–128.

Storelvmo, T., Kristjánsson, J. E., Lohmann, U., Iversen, T., Kirkevåg, A. and Seland, O. 2008. Modeling of the Wegener-Bergeron-Findeisen Process-Implications for Aerosol Indirect Effects. *Environmental Research Letters 3*(4), p. 045001 (10pp).

Stuppacher, I. and Supan, P. 2007. Rendering of Water Drops in Real-Time. In *CESCG 2007: The 11th Central European Seminar on Computer Graphics.* Available from: `http://www.cg.tuwien.ac.at/`

hostings/cescg/CESCG-2007/papers/Hagenberg-Stuppacher-Ines/cescg_StuppacherInes.pdf [Accessed 17 Dec 2008].

Sun, B., Ramamoorthi, R., Narasimhan, S. G. and Nayar, S. K. 2005. A Practical Analytic Single Scattering Model for Real Time Rendering. *ACM Trans. Graph. 24*(3), pp. 1040–1049.

Sundog Software. ca.2008. Silverlining by Sundog Software. Available from: http://www.sundog-soft.com/ [Accessed 17 Dec 2008].

Tariq, S. 2007. Rain - NVIDIA Direct3D SDK 10 Technical Demonstration.

Tatarchuk, N. and Isidoro, J. 2006. Artist-Directable Real-Time Rain Rendering in City Environments. In *Proceedings of Eurographics Workshop on Natural Phenomena*, ACM, New York, NY, USA.

Trembilski, A. and Brossler, A. 2002a. Surface-based Efficient Cloud Visualisation for Animation Applications. In *Proceedings of The 10-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision'2002*, WSCG, pp. 453–460.

Trembilski, A. and Brossler, A. 2002b. Transparency for Folygon Based Cloud Rendering. In *SAC '02: Proceedings of the 2002 ACM symposium on Applied computing*, ACM, New York, NY, USA, pp. 785–790.

Treuille, A., McNamara, A., Popović, Z. and Stam, J. 2003. Keyframe Control of Smoke Simulations. In *ACM SIGGRAPH 2003*, ACM, New York, NY, USA, pp. 716–723.

Vane, E. 2004. Cloud Rendering using 3D Textures. Computer Graphics Lab, University of Waterloo. Available from: http://serv1.ist.psu.edu:8080/viewdoc/download;jsessionid=BF3ECEBA5D5618335748717423C5A645?doi=10.1.1.96.4322&rep=rep1&type=pdf [Accessed 16 Dec 2008].

Venere, E. 2005. Purdue student helps movie animators take clouds to new highs. Purdue University News Report. Available from: http://www.purdue.edu/UNS/html3month/2005/050413.Ebert.Valiant.html [Accessed 17 Dec 2008].

Vis5D. ca.2012. Project website. Available from: http://vis5d.sourceforge.net/ [Accessed 16 July 2012].

Wang, N. and Wade, B. 2005. Let it snow, let it snow, let it snow (and rain). In *Game Programming Gems 5*, K. Pallister, Ed. Charles River Media, pp. 507–513.

Wang, L., Lin, Z., Fang, T., Yang, X., Yu, X. and Kang, S. B. 2006. Real-Time Rendering of Realistic Rain. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Sketches*, ACM, New York, NY, USA, p. 156.

Wang, N. 2004. Realistic and Fast Cloud Rendering. *Journal of Graphics Tools 9*(3), pp. 21–40.

Wenzel, C. 2006. Real-time atmospheric effects in games. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Courses*, ACM, New York, NY, USA, pp. 113–128.

West, M. 2008. Practical Fluid Dynamics: Part 1. Available from: `http://www.gamasutra.com/view/feature/1549/practical_fluid_dynamics_part_1.php` [Accessed 15 Jun 2013].

Wilhelmson, B. and Ramamurthy, M. ca.2008. Weather World 2010: Fair Weather Cumulus Clouds. Department of Atmospheric Sciences, University of Illinois at Urbana-Champaign. Available from: `http://ww2010.atmos.uiuc.edu/(Gh)/guides/mtr/cld/cldtyp/vrt/frcu.rxml` [Accessed 15 Dec 2008].

WMO. 1975. *International Cloud Atlas.* World Meteorological Organisation, Geneva, Switzerland.

Xue, M., Droegemeier, K. and Wong, V. 2000. The Advanced Regional Prediction System (ARPS) - A multi-scale nonhydrostatic atmospheric simulation and prediction model. Part I: Model dynamics and verification. *Meteorology and Atmospheric Physics 75*, pp. 161–193.

Xue, M., Droegemeier, K., Wong, V., Shapiro, A., Brewster, K., Carr, F., Weber, D., Liu, Y. and Wang, D. 2001. The Advanced Regional Prediction System (ARPS) - A multi-scale nonhydrostatic atmospheric simulation and prediction model. Part II: Model physics and applications. *Meteorology and Atmospheric Physics 76*, pp. 143–165.

Yaeger, L., Upson, C. and Myers, R. 1986. Combining Physical and Visual Simulation—Creation of the Planet Jupiter for the Film “2010”. *SIGGRAPH Comput. Graph. 20*(4), pp. 85–93.

Zafar, N. B., Akesson, J., Roble, D. and Museth, K. 2006. Scattered Spherical Harmonic Approximation for Accelerated Volume Rendering. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Sketches*, ACM, New York, NY, USA, p. 148.

Zhou, Y., Shi, J. and Yu, J. 2006. Free and Shape-controlled Flows of Smoke. In *VRCIA '06: Proceedings of the 2006 ACM international conference* *on Virtual reality continuum and its applications*, ACM, New York, NY, USA, pp. 105–112.

Appendix A

Fluid Dynamics

This section provides an overview of the main concepts involved in fluid dynamics, by introducing it from the point of view of computer graphics. This is an important distinction, for the goals of CG are quite different to those of meteorology, and thus impact on the solution to such methods. While in meteorology the goal is to provide as accurate a simulation as possible, in CG we are only interested in visual believability — whether it is achieved through a physically-based simulation or not is largely irrelevant to our goals. This is especially true of fluid dynamics, where it is nearly impossible to tell if the intricate flows are completely accurate, or the result of a talented artist.

The subject of fluid dynamics is vast and is not the main topic of this work, so solutions to the equations will only be reviewed briefly, and a very small range of techniques will be looked at. Further, the explanation for the equations themselves is given to provide an intuitive understanding, rather than a rigorous mathematical proof. It is felt that this is potentially more valuable, and should prove more useful when other works are examined. Of course references will be given, where appropriate, so that the interested reader may pursue individual topics further. Of particular interest is the work by Bridson [2008], which is specifically directed at fluid simulation for computer graphics (a statement which is literally the name of the book).

A.1 Mathematics Review

Before covering fluid dynamics, it is worth taking a moment to review some of the mathematics that will be used. Specifically, those involving vector fields.

A.1.1 Vector Fields

If we consider a small volume of three-dimensional space, we can sample it at evenly-spaced points. At each of these sampling points, we can examine some property of the volume that results in a vector which points in some direction or other. This is the essence of a vector field. The field itself is continuous though, and the sampling points are just to aid visualisation. Vector fields have a number of properties that are fundamental to fluid dynamics.

A.1.1.1 A Word on Notation: The Del Operator

As an item of pure convenience, we can use an inverted Greek letter delta to mean a vector of partial derivatives:

$$\nabla = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z} \right)$$

This symbol is known as the *nabla* or *del* operator.

A.1.1.2 Gradient

The *gradient* is actually a property of a scalar field rather than a vector field, but it results in a vector. It is simply the rate of change of the scalar field's values, in each dimension of the field. Mathematically, then, it is defined as:

$$grad\, v = \nabla v = \left(\frac{\partial v}{\partial x}, \frac{\partial v}{\partial y}, \frac{\partial v}{\partial z} \right)$$

A point to note is that the gradient is positive if the scalar field increases in the positive axis direction.

A.1.1.3 Divergence

The divergence of a vector field is a measure of how much the field is expanding. For example if all the vectors are pointing away from the origin the field has positive divergence. This results in a scalar value, and is defined as:

$$div\, \mathbf{v}(v_x, v_y, v_z) = \nabla \cdot \mathbf{v} = \frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} + \frac{\partial v_z}{\partial z}$$

A.1.1.4 Curl

At each of the sample points in the vector field, consider that they have the ability to rotate on the spot, depending on the 'forces' exerted on them by the vectors. This microscopic rotation is the *curl*.

In three-dimensions it is defined as:

$$curl\,\mathbf{v}(v_x, v_y, v_z) = \nabla \times \mathbf{v} = \left(\frac{\partial v_z}{\partial y} - \frac{\partial v_y}{\partial z}, \frac{\partial v_x}{\partial z} - \frac{\partial v_z}{\partial x}, \frac{\partial v_y}{\partial x} - \frac{\partial v_x}{\partial y}\right)$$

In two-dimensions, the curl of a vector field results in a scalar field:

$$curl\,\mathbf{v}(v_x, v_y) = \nabla \times \mathbf{v} = \frac{\partial v_y}{\partial x} - \frac{\partial v_x}{\partial y}$$

While the curl of a scalar field results in a vector field:

$$curl\,v = \nabla \times v = \left(\frac{\partial v}{\partial x}, -\frac{\partial v}{\partial y}\right)$$

A.1.1.5 The Laplace Operator

This is defined as the "divergence of the gradient" [Bridson 2008, Appendix A]. It results in a scalar and in three dimensions is expressed as:

$$\triangle v = \nabla^2 v = \nabla \cdot \nabla v = \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 v}{\partial z^2}$$

A.2 Equations of Fluid Dynamics

The motions of an incompressible fluid can be described by the following two equations:

$$\frac{D\mathbf{u}}{Dt} = -\frac{1}{\rho}\nabla p + v\nabla^2 \mathbf{u} \tag{A.1}$$

$$\nabla \cdot \mathbf{u} = 0$$

The first of these is the Navier-Stokes *momentum* equation, and the second is the *mass continuity* equation. We shall look briefly at how they are built up, before tackling the task of solving them.

A.2.1 The Momentum Equation

This equation is actually the result of Newton's second law of motion, the equation $F = ma$. Let us consider a fluid that is described by a number of particles, or 'dyed blobs' if you prefer [Acheson 1990]. We are trying to describe its motion, so we need to find a formula for the particle's acceleration. Since the mass of the particle will remain constant during its motions, all we need to do is to find the forces that affect it. We can write this as:

$$m\frac{D\mathbf{u}}{Dt} = F \tag{A.2}$$

where we express the acceleration of the particle as the change in its velocity vector, **u**, over time using 'large D' notation, which defines the *material derivative* (the actual meaning of which we shall look at later).

A.2.2 Pressure

If we ignore for the moment that we have stated that our fluid is incompressible, the first force which we need to consider is that which results from the other particles in the fluid: pressure. If there is a pressure acting on one side of our particle, and another pressure acting on the other side, we would assume that the particle would move towards the lower pressure. Specifically, then, we need to find the force that results from the *difference* in pressure across the particle. We have already looked at a very similar problem, when we used hydrostatics to describe the pressure gradient of the atmosphere, in Section 2.3.1.2 on page 32. This time, however, we need to consider pressure in all directions rather than just the height. Let us start by abstracting our particle as a cube:

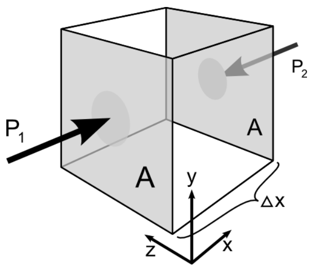


Figure A.1: Pressures acting on two sides of a 'cube of fluid'.

From Figure A.1 we can describe two forces acting against the particle on the x-axis:

$$\begin{aligned} F_1 &= P_1 A \\ F_2 &= -P_2 A \end{aligned}$$

Notice the sign-change in the second force. This is because the force acts towards our cubic particle, but is in opposite direction to our x-axis. The resultant force from these two is therefore:

$$\begin{aligned}
F_x &= F_1 + F_2 \\
&= P_1 A - P_2 A \\
&= -(P_2 - P_1)A \\
&= -\Delta P.A
\end{aligned}$$

Here, we assume that our definition of ΔP, the change in pressure across the particle, acts in the 'wrong direction', i.e. towards a greater value of pressure, while our force needs to act towards a lesser value in pressure. The reasoning for this is that we are eventually going to want to use a differential change in pressure, and this is the direction that is described by the gradient.

Our force currently involves the area of one side of our cubic particle, which is not particularly useful. We can use Newton's third law and a definition of the volume of our particle to help rearrange this:

$$\begin{aligned}
F_x &= -\Delta P.A \\
ma_x &= -\Delta P.A \\
(V\rho)a_x &= -\Delta P.A \\
(\Delta x.A\rho)a_x &= -\Delta P.A \\
a_x &= -\frac{1}{\rho}.\frac{\Delta p}{\Delta x} \\
F_x &= -\frac{m}{\rho}.\frac{\Delta p}{\Delta x}
\end{aligned}$$

We now have a useful format, and all that remains to be done now is to consider what happens as Δx becomes very small:

$$F_x = -\frac{m}{\rho}.\frac{\partial p}{\partial x}$$

And finally, the full three-dimensional force can be considered:

$$\begin{aligned}
F &= -\frac{m}{\rho}\left(\frac{\partial p}{\partial x}, \frac{\partial p}{\partial y}, \frac{\partial p}{\partial z}\right) \\
F &= -\frac{m}{\rho}\nabla p
\end{aligned}$$

where the last arrangement shows the definition of the gradient of a vector.

We are now in a position to add this as a force acting on our particle, using our initial momentum equation A.2.

$$\begin{aligned} m\frac{D\mathbf{u}}{Dt} &= F \\ m\frac{D\mathbf{u}}{Dt} &= -\frac{m}{\rho}\nabla p \\ \frac{D\mathbf{u}}{Dt} &= -\frac{1}{\rho}\nabla p \end{aligned}$$

This should be starting to look familiar, since it is now a subset of the Navier-Stokes momentum equation, A.1 on page 141. The final term is the resistive force of *viscosity*. As it stands, our momentum equation is already useful, and some fluids can be described to a reasonable degree of accuracy without a viscosity component. Such a fluid is known as *inviscid*, and the momentum equation is *Euler's equation of inviscid flow*. Let us continue for the moment, though, and look at the viscosity component.

A.2.3 Viscosity

Within a viscous fluid, a fast moving particle is slowed down, indicating that a force must act upon it. This force is a result of interactions with the other particles in the fluid, and acts to change the particle's velocity by diffusing it with its neighbours. From this definition[1], we can intuitively build up a simple equation for viscosity[2]:

Recall that the *Laplacian* operator (∇^2) gives us a measure of how different a quantity is from the average around it. If we assume that the amount of viscosity is constant, which is not always the case, then we can modulate the force using a *viscosity coefficient*. Applying the Laplacian operator to the fluid velocity, and using the viscosity coefficient, we obtain the force due to viscosity quite simply:

$$F = v\nabla^2\mathbf{u}$$

A.2.4 Other Forces

It is worth bearing in mind that additional forces can easily be added to this momentum equation. From the work in the previous chapters, one force that should spring to mind is that of buoyancy. Another that is commonly used in large-scale

[1]To reiterate the statement from the introduction to this chapter, the intention here is to provide more of an *explanation* than a rigorous mathematical proof, since proofs are often unintuitive to non-experts and can always be found in a good textbook.

[2]This explanation is paraphrased from Bridson [2008, p. 5].

meteorological situations is that of the *Coriolis* force, which results from movements within the rotating frame of reference caused by the Earth spinning on its axis.[3] Within computer graphics, it is common to simply add forces that result from artistic or user input, as a means of directing and controlling the flow. All such forces are usually combined into a single term, **g**, such that the momentum equation becomes:

$$\frac{D\mathbf{u}}{Dt} = -\frac{1}{\rho}\nabla p + v\nabla^2\mathbf{u} + \mathbf{g}$$

A.2.5 Mass Continuity: Pressure in an Incompressible Fluid?

We have defined our fluid as incompressible, which is actually a surprisingly good approximation for fluids that move slower than the speed of sound (i.e. less than approximately 300m/s). From this definition, we can deduce that the fluid must not be able to 'bunch-up' or 'spread-out', since this would involve compression and expansion.

Recall that the divergence operator gives us a measure of how much a vector field is expanding or contracting. We can therefore intuitively derive the mass continuity equation in one step, since we are expecting no divergence in the velocities of the fluid:

$$\nabla \cdot \mathbf{u} = 0$$

Such a fluid is termed *divergence free* and what this also implies is that density must remain constant with time. The only remaining conundrum is that of our pressure term, the perturbations of which are surely a result of changes in density. Here, the best explanation is that we must loosen our definition of 'pressure' to be that it is "whatever it takes to keep the velocity divergence-free" [Bridson 2008, p. 12].

A.2.6 Lagrangian and Eulerian Viewpoints

So far, we have considered our fluid to comprise of a number of particles in what is known as a *Lagrangian* viewpoint. However, there are issues with this line of thinking — for instance, evaluating the pressure gradient in a group of particles can become difficult.

[3]The Coriolis force is small enough that it can safely be ignored at scales of a few tens of kilometres, but is an important factor in planet-scale circulations, and has some interesting consequences. For more information, the reader is directed to standard texts.

An alternative viewpoint also exists, in which the volume of liquid is discretised into a lattice, or grid. This is the *Eulerian* perspective.

The two viewpoints are linked mathematically through the *material derivative*, also called the *total derivative*, which results in the 'large D' notation. If we consider a particle, its position can be thought of as a function of time. If the particle has a property, q, then from the Lagrangian perspective we just need to query the particle's current state. From the perspective of a single grid-cell in the Eulerian point of view, however, the value of q changes as various different particles move through its bounds. To evaluate q from this viewpoint, we need to take into account the movements of the particles that flow past, so we can treat q as a function of position and time, i.e. $q(x(t), y(t), z(t), t)$. We are now in a position to relate the rate of change of q in both perspectives:

$$\begin{aligned}\frac{Dq}{Dt} &= \frac{d}{dt}q(x(t), y(t), z(t), t)\\ &= \frac{\partial q}{\partial x}\frac{dx}{dt} + \frac{\partial q}{\partial y}\frac{dy}{dt} + \frac{\partial q}{\partial z}\frac{dz}{dt} + \frac{\partial q}{\partial t}\\ &= \frac{\partial q}{\partial t} + u\frac{\partial q}{\partial x} + v\frac{\partial q}{\partial y} + w\frac{\partial q}{\partial z}\\ &= \frac{\partial q}{\partial t} + (\mathbf{u}\cdot\nabla)q\end{aligned}$$

Here, we used the chain rule (i.e. $\frac{d}{dx}(f(g(x))) = f'(g(x))g'(x)$), and defined the fluid velocity as $\mathbf{u}(u, v, w)$, to obtain the definition of the material derivative. The property of the fluid that we are interested in is its velocity, and we can easily put this into the equation:

$$\frac{D\mathbf{u}}{Dt} = \frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u}$$

This is also known as the *self-advection* term. In terms of the notation (which Bridson [2008, p. 10] recognises as a little unintuitive), when using a vector property each component should be treated separately. Thus, for our fluid velocity $\mathbf{u}(u, v, w)$, we have: [Bridson 2008, p. 10]

$$\frac{D\mathbf{u}}{Dt} = \begin{bmatrix}\frac{Du}{Dt}\\ \frac{Dv}{Dt}\\ \frac{Dw}{Dt}\end{bmatrix} = \begin{bmatrix}\frac{\partial u}{\partial t} + (\mathbf{u}\cdot\nabla)u\\ \frac{\partial v}{\partial t} + (\mathbf{u}\cdot\nabla)v\\ \frac{\partial w}{\partial t} + (\mathbf{u}\cdot\nabla)w\end{bmatrix} = \frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u}$$

Finally, we can express the Navier-Stokes momentum equation from an Eulerian viewpoint as:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u} = -\frac{1}{\rho}\nabla p + v\nabla^2\mathbf{u}$$

A.2.7 Boundary Conditions

We have now intuitively derived the Navier-Stokes equations, and we are nearly ready to consider how to solve them. The universe, however, is not neatly composed of one continuous fluid, an observation which leads us to the conclusion that interfaces must exist between fluids and other objects, be they solid or different fluids. Further, computational resources and power are (unfortunately) finite, which means that even for a large fluid, we are unlikely to be able to simulate it in its entirety.

In the case of a fluid interfacing with a solid object, we have a number of options. We can enforce the normal component of the fluid's velocity to be zero, so that instead of hitting the object, it slides over it. This is the *no-stick* condition, a name which implies its opposite, which is actually called the *no-slip* condition, in which all velocity components of the fluid are set to zero. What this results in is a region of the fluid near to the object in which the average velocity gradually reduces as the object is approached. Such a region is called a *boundary layer*. This is the more physically realistic of the two approaches, and has some interesting consequences, such as turbulent wakes behind objects in a moving flow (or moving objects in a mostly stationary fluid). [Acheson 1990, p. 26]

At our simulation boundaries, also called *free surfaces*, we have a number of options. We could treat the bounds as solid walls, and apply either the no-stick or no-slip conditions. Alternatively, in a condition especially useful for generating tilable textures, we can define a *periodic* boundary, where flow that leaves one side enters the opposite. [Bridson 2008]

A.3 2D Vortices

One of the features of fluid flow that is so visually captivating is its ability to form rotating swirls of motion, often with tragically devastating consequences, as demonstrated by tornadoes. It may not be surprising to find, then, that the fluid dynamics equations can be expressed in terms of *vortices*. Here, we shall examine the two-dimensional case, and briefly look at issues when describing the three-dimensional case.

A.3.1 Definition of vorticity

Let us consider a point in our fluid, and fix in it a 'measurement' device that is free to rotate but not to move. As the fluid flows past our device, it applies a force to 'vanes' on either side of the central axis. Naturally, if the force is greater on one side than the other, our measurement device will register a rotation.

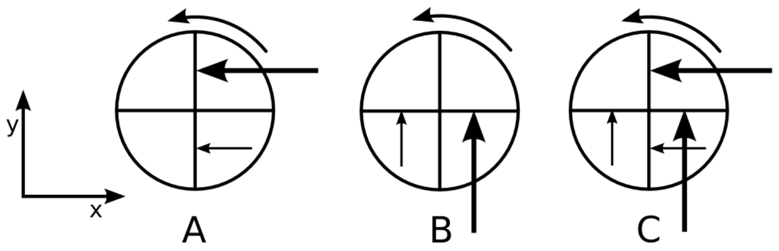


Figure A.2: Adapted from Griebel et al. Fig 4.6 [1997, p. 60].

Let us consider what happens if the device spins anticlockwise. The forces acting in the x-axis must be as in Figure A.2a, which tells us that the velocity of the fluid 'negatively increases' in the positive y-axis direction. Let us denote the x-axis velocity as u. Similarly, in Figure A.2b we can see that the y-axis velocity component, which we shall term v, increases in the x-axis direction. The rotational property we have described is called vorticity and, for our two-dimensional model, is defined as:

$$\omega = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}$$

It is important to note that vorticity is a 'microscopic' property of how much the fluid is rotating at a very small point. From this definition and our mathematical definition above, we can see that it can also be related to vector fields — recall that the definition of curl in two-dimensions is:

$$curl\,\mathbf{v}(v_x, v_y) = \nabla \times \mathbf{v} = \frac{\partial v_y}{\partial x} - \frac{\partial v_x}{\partial y}$$

Thus, we can define vorticity as the curl of the velocity field:

$$\omega = \nabla \times \mathbf{u}$$

This also extends into the three-dimensional case, but here the curl of the velocity results in a vector, which can lead to complications when we come to solve the equations.

A.3.2 Expressing the Navier-Stokes Equations using Vorticity

If we start with the Navier-Stokes equations from the Eulerian viewpoint, we have:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u} = -\frac{1}{\rho}\nabla p + v\nabla^2\mathbf{u}$$

Our definition of vorticity is that it is the curl of the velocity, so we need to introduce curl into the equation. The easiest way of doing this is simply to take the curl of both sides: [Bridson 2008, pp. 128-129]

$$\begin{aligned}
\nabla \times \frac{\partial \mathbf{u}}{\partial t} + \nabla \times (\mathbf{u} \cdot \nabla)\mathbf{u} &= -\nabla \times \frac{1}{\rho}\nabla p + \nabla \times v\nabla^2 \mathbf{u} \\
\frac{\partial \nabla \times \mathbf{u}}{\partial t} + \nabla \times (\mathbf{u} \cdot \nabla)\mathbf{u} &= -\frac{1}{\rho}\nabla \times \nabla p + v\nabla^2(\nabla \times \mathbf{u}) \\
\frac{\partial \omega}{\partial t} + \nabla \times (\mathbf{u} \cdot \nabla)\mathbf{u} &= v\nabla^2 \omega \qquad \text{(A.3)}
\end{aligned}$$

where we used the fact that the curl of a gradient is zero to remove the pressure term. Ideally, we would like to rearrange the second term into something more useful (but unfortunately this is something of a backwards step, because it won't become clear why it is useful until we get there). We can start by rearranging a known identity:

$$\begin{aligned}
\nabla(\mathbf{A} \cdot \mathbf{B}) &\equiv \mathbf{A} \times (\nabla \times \mathbf{B}) + (\mathbf{A} \cdot \nabla)\mathbf{B} + \mathbf{B} \times (\nabla \times \mathbf{A}) + (\mathbf{B} \cdot \nabla)\mathbf{A} \\
(\mathbf{A} \cdot \nabla)\mathbf{B} &= \nabla(\mathbf{A} \cdot \mathbf{B}) - \mathbf{A} \times (\nabla \times \mathbf{B}) - \mathbf{B} \times (\nabla \times \mathbf{A}) - (\mathbf{B} \cdot \nabla)\mathbf{A}
\end{aligned}$$

We can then use this on **u**:

$$\begin{aligned}
(\mathbf{u} \cdot \nabla)\mathbf{u} &= \nabla(\mathbf{u} \cdot \mathbf{u}) - \mathbf{u} \times (\nabla \times \mathbf{u}) - \mathbf{u} \times (\nabla \times \mathbf{u}) - (\mathbf{u} \cdot \nabla)\mathbf{u} \\
2(\mathbf{u} \cdot \nabla)\mathbf{u} &= \nabla(\mathbf{u}^2) - 2\mathbf{u} \times (\nabla \times \mathbf{u}) \\
(\mathbf{u} \cdot \nabla)\mathbf{u} &= \nabla\left(\frac{1}{2}\mathbf{u}^2\right) - \mathbf{u} \times (\nabla \times \mathbf{u})
\end{aligned}$$

which, remembering that $\mathbf{A} \times \mathbf{B} \equiv -\mathbf{B} \times \mathbf{A}$, we can also write as:

$$(\mathbf{u} \cdot \nabla)\mathbf{u} = \nabla\left(\frac{1}{2}\mathbf{u}^2\right) + (\nabla \times \mathbf{u}) \times \mathbf{u}$$

If we take the curl of this:

$$\nabla \times (\mathbf{u} \cdot \nabla)\mathbf{u} = \nabla \times \nabla\left(\frac{1}{2}\mathbf{u}^2\right) + \nabla \times ((\nabla \times \mathbf{u}) \times \mathbf{u})$$

But, the curl of a gradient is always zero, so we can remove the first term on the right. If we also use our definition of vorticity on the right side, we can re-express this:

$$\begin{aligned}\nabla \times (\mathbf{u} \cdot \nabla)\mathbf{u} &= -\nabla \times ((\nabla \times \mathbf{u}) \times \mathbf{u}) \\ \nabla \times (\mathbf{u} \cdot \nabla)\mathbf{u} &= -\nabla \times (\omega \times \mathbf{u})\end{aligned}$$

The term on the right now takes the recognisable form of another identity:

$$\nabla \times (\mathbf{A} \times \mathbf{B}) \equiv (\mathbf{B} \cdot \nabla)\mathbf{A} + \mathbf{A}(\nabla \cdot \mathbf{B}) - (\mathbf{A} \cdot \nabla)\mathbf{B} - \mathbf{B}(\nabla \cdot \mathbf{A})$$

Which we can now apply:

$$\begin{aligned}\nabla \times (\mathbf{u} \cdot \nabla)\mathbf{u} &= \nabla \times (\omega \times \mathbf{u}) \\ &= (\mathbf{u} \cdot \nabla)\omega + \omega(\nabla \cdot \mathbf{u}) - (\omega \cdot \nabla)\mathbf{u} - \mathbf{u}(\nabla \cdot \omega)\end{aligned}$$

We can immediately remove a couple of terms, since our mass continuity equation states that $\nabla \cdot \mathbf{u} = 0$, and if we expand: $\nabla \cdot \omega = \nabla \cdot \nabla \times \mathbf{u}$, we can use the fact that the gradient of the curl is zero, to obtain:

$$\nabla \times (\mathbf{u} \cdot \nabla)\mathbf{u} = (\mathbf{u} \cdot \nabla)\omega - (\omega \cdot \nabla)\mathbf{u}$$

We can then substitute this into equation A.3, to obtain:

$$\begin{aligned}\frac{\partial \omega}{\partial t} + (\mathbf{u} \cdot \nabla)\omega - (\omega \cdot \nabla)\mathbf{u} &= v\nabla^2\omega \\ \frac{\partial \omega}{\partial t} + (\mathbf{u} \cdot \nabla)\omega &= (\omega \cdot \nabla)\mathbf{u} + v\nabla^2\omega \\ \frac{D\omega}{Dt} &= (\omega \cdot \nabla)\mathbf{u} + v\nabla^2\omega\end{aligned}$$

In this last step we can see the fruits of our labour, since our rearrangements gave us the material derivative, which we are then free to use to switch to the Lagrangian viewpoint. This is the vortex form of the Navier-Stokes equation, also called the *vorticity equation* [Bridson 2008, p. 129]. The first term on the right side is known as the *vorticity-stretching* term, and is another factor that must be dealt with in three-dimensional flows. Thankfully, this term completely disappears in two-dimensions. This can be thought through by assuming that the two-dimensional case is actually a slice through three-dimensional space. In this case, the vorticity vector becomes zero in the x and y values, while the velocity vector is zero in the z direction. The dot product between them then results in zero, i.e. $\omega(0, 0, \omega) \cdot \mathbf{u}(u, v, 0) = 0$. [Bridson 2008, p. 129]

In two-dimensions, then, we can express the vorticity equation as:

$$\frac{D\omega}{Dt} = v\nabla^2\omega$$

So the change in vorticity for a 'vortex particle' is only dependent on viscosity. Thus, it follows that for an inviscid flow:

$$\frac{D\omega}{Dt} = 0$$

The appeal behind vortex techniques should now be obvious: there is no need to worry about the pressure term at all. We have, however, neglected our additional forces, such as buoyancy. These can be easily added to the equation by taking the curl of the term.

A.4 Solving The Equations

The problem of solving the fluid dynamics equations is a topic of ongoing research, and solutions can vary widely in approach and complexity. The approach taken here is to provide an overview of how the problem is solved, with the goal of identifying general approaches and to facilitate discussion.

A.4.1 Solving the Eulerian Navier-Stokes Equations

Here we shall look at approaches to solving the equations in relation to a finite volume that is discretised into a grid, or lattice. In order to deal with properties of individual cells, we need to briefly introduce some notation[4]. A property q, at grid cell i in a one-dimensional simplification, at time-step n, shall be noted as: q_i^n, where it must be remembered that the superscript is *not* raising q to a power.

The general approach is that for each grid cell, the fluid properties will be updated based on the solution of the equations from one discrete time-step to another. In an offline simulation, the time-step can be the time between individual frames, or possibly smaller if conditions require it. In an interactive 'real-time' system, the time-step will usually be the time between frames, or some multiple thereof. Alternatively, the simulation update can be decoupled from the rendering process, and a single simulation update spread over several frames, which can be useful because solving the equations is computationally expensive. Either way, the time-step chosen can be an important factor in how the equations are solved.

[4]We shall use the notation presented by Bridson [2008] since, as he explains, it is that which is most commonly used in computer graphics fluid simulation.

A.4.2 Splitting the Equations

The equations themselves are fairly involved, so the first task is to split them into component parts, each of which may then be solved separately. This is usually done as follows:

$$\begin{aligned}\frac{Dq}{Dt} &= 0\\ \frac{\partial \mathbf{u}}{\partial t} &= \mathbf{g}\\ \frac{\partial \mathbf{u}}{\partial t} &= -\frac{1}{\rho}\nabla p\\ while\, ensuring\, that:\ \nabla \cdot \mathbf{u} &= 0\end{aligned}$$

The first of these is the general advection term, which is almost hidden in the Lagrangian form of the equations, since property conservation is a trivial problem for a particle — we just have to remember not to change its properties. The second equation describes the *body forces*, and finally we have the pressure/incompressibility equations. [Bridson 2008, p. 20]

The idea in splitting is that we can then solve the simpler equations separately, and combine the results. At first glance, the combination may appear to be a simple summation of the results, but unfortunately this is not *quite* the case, for the order of operations can also be important. The advection equation solution is usually expected to operate on a divergence-free vector field, but thankfully this is exactly what the pressure/incompressibility equation outputs. All we need to do, then, is to ensure that advection is the first operation performed, since we can assume that the output of the previous iteration was a divergence-free vector field.

Note that we have actually split the Euler equations, since viscosity is absent. If it is required, this can indeed also be separated out. The solution for viscosity is similar to that for pressure, so will be discussed at this point.

Let us now look at various approaches at solving each of these equations.

A.4.3 Advection

Recall that the purpose of this step is to transport properties of the fluid from one grid cell to another, based on a velocity vector field.

If we start with the equation for advecting a general property, q, in just a single dimension, we have:

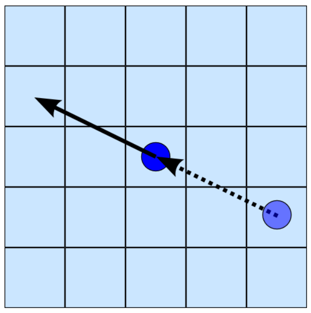


Figure A.3: In Stam's advection scheme [Stam 1999] the cell's velocity vector is traced backwards. The fluid property is then sampled at this point and brought forwards to the current cell.

$$\begin{aligned} \frac{Dq}{Dt} &= 0 \\ \frac{\partial q}{\partial t} + u\frac{\partial q}{\partial x} &= 0 \end{aligned}$$

where u is the velocity along the axis of our one-dimensional model. We could then replace the differential quantities with finite differences: [Bridson 2008, p. 27]

$$\begin{aligned} \frac{q_i^{n+1} - q_i^n}{\Delta t} + u_i^n \frac{q_{i+1}^n - q_{i-q}^n}{2\Delta x} &= 0 \\ q_i^{n+1} &= q_i^n - \Delta t u_i^n \frac{q_{i+1}^n - q_{i-q}^n}{2\Delta x} \end{aligned}$$

We could leave it there, but this equation has a property that can cause problems: it is completely *unstable*. This means that the fluid literally appears to 'blow-up', especially when the time-step is insufficiently small (as was hinted at in the beginning of this section). As a solution, we can use the *semi-Lagrangian* technique described by Stam [1999], which is inherently *stable* by nature for any time-step, and as such is highly suited to real-time applications.

Stam's technique is simple but extremely effective: rather than moving a property forwards along the line of velocity, it works by pretending that the current cell contains a particle, and applies the reverse velocity to trace where it came from. It then takes the property from this sample point, moving it forwards to the current cell. (See Figure A.3.)

Mathematically, this could be defined as[5]:

[5]This definition uses forward Euler integration, which is not necessarily the best technique. Bridson [2008, p. 29 and Appendix A] discusses alternatives.

$$q_{i,j}^{n+1} = q_{(i,j)-\Delta t u_{i,j}^{n}}^{n}$$

However, since we have discretised our spatial domain into a finite number of grid-cells, as per the Eulerian approach, it is highly unlikely that tracing the velocity backwards would result in a sampling point that is exactly at the centre of a cell. This means that we must interpolate between cells to acquire the sampling value. Unfortunately, this introduces another problem: since we are essentially forced to perform an averaging operation, any quantity we advect will eventually dissipate over time. Recall, though, that we are trying to advect the velocity itself, which means that this too will dissipate. Intuitively, we can relate this to our definition of viscosity, and mathematically it turns out that this is very similar [Bridson 2008, p. 36]. Since real-world fluids are viscous, this may not be a problem, but methods do exist that can minimise this effect [Bridson 2008].

At locations near the boundaries of the fluid volume, the backward tracing may result in samples that are outside the simulation domain. The operation that must be performed in this case is then dependent on the boundary condition. For instance, in a periodic boundary, the sampling coordinates are simply wrapped around so that the sample is taken from the opposite side.

A complementary technique is that of *accounting advection,* in which mass-conservation is handled in the advection solution [West 2008, p. 3]. This means that the advection solution will conserve mass independently of whether the vector field does or not. This technique extends either the forward or backward advection described above by simply subtracting the advected quantity from the source cell(s) when it is copied to its destination. The result is no net loss or gain in the advected property.

It should be noted that at this stage any other properties of the the simulation, such as water mixing-ratios and potential temperature, must also be advected in the same manner as the velocity.

A.4.4 Body Forces

This is potentially the easiest term to solve, and may be approximated to a good degree by the use of forward Euler integration: [Bridson 2008, p. 20]

$$\mathbf{u}_i^{n+1} = \mathbf{u}_i^{n} + \Delta t\mathbf{g}$$

Note, however, that the ‘previous velocity’ vector (i.e. $\mathbf{u}_i^n$) will actually be the temporary output from the previous operation.

A.4.5 Pressure/Incompressibility and Viscosity

We can assume that the previous steps resulted in a velocity vector field that is not divergence-free. The purpose of this step is to produce an 'adjustment' which ensures that the velocity vector field is divergence free. To do this, we can use the *Hodge-Helmholtz decomposition*, which states that a vector field, **A**, can be described as the sum of a divergence-free vector field, **B**, and the gradient of a scalar field, C:

$$\mathbf{A} = \mathbf{B} + \nabla C$$

In our case, **A** is the output from the previous steps, **B** is the vector field we are trying to find, and C is the adjustment which we call 'pressure'. Thus, if we label our input velocity vector field **A** to distinguish it from our desired divergence-free field **u**, we can express this as:

$$\mathbf{u} = \mathbf{A} - \nabla p \tag{A.4}$$

where p is our pressure term. If we take the divergence of each side, we obtain:

$$\nabla \cdot \mathbf{u} = \nabla \cdot \mathbf{A} - \nabla^2 p$$

However, we know that $\nabla \cdot \mathbf{u} = 0$, so we now have a means of expressing the pressure in terms of our input vector field:

$$\nabla^2 p = \nabla \cdot \mathbf{A}$$

This is a *Poisson equation* and, thankfully, a whole myriad of techniques exist for solving them to give us p. One of the simplest is *Jacobi iteration*, which slowly converges on the answer. The general form for Jacobi iteration in two-dimensions is: [Harris 2004]

$$x_{i,j}^{(k+1)} = \frac{x_{i-1,j}^{(k)} + x_{i+1,j}^{(k)} + x_{i,j-1}^{(k)} + x_{i,j+1}^{(k)} + \alpha b_{i,j}}{\beta}$$

In the case of the pressure equation above, we define the variables and constants as follows: [Harris 2004]

$$\begin{aligned} x &= p \\ \alpha &= -(\Delta x)^2 \\ b &= \nabla \cdot \mathbf{A} \\ \beta &= 4 \end{aligned}$$

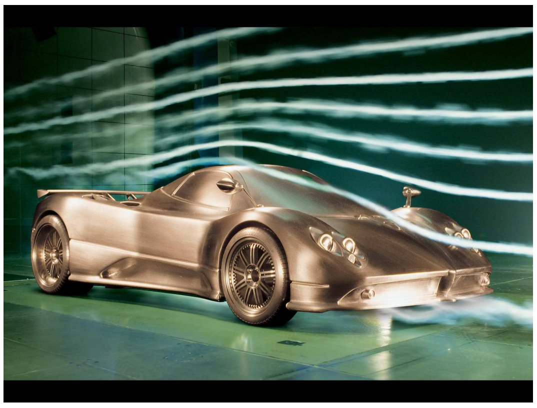


Figure A.4: A car in a wind tunnel, with tracer smoke to show streamlines in the fluid flow. Image credited to Pagani [2005]

Where Δx is the size of a grid cell. Thus armed with the pressure field, we can subtract its gradient from our input vector field, as per equation A.4, to obtain the final divergence-free velocity vector field.

Of course, we may not be simulating an inviscid fluid, so finally we need to consider the viscosity equation. This is also a Poisson equation, and can be solved in the same way as the pressure term. Using the above Jacobi technique, we can define the variables and constants as follows: [Harris 2004]

$$\begin{aligned} x &= \mathbf{u} \\ \alpha &= \frac{(\Delta x)^2}{n\Delta t} \\ b &= \mathbf{A} \\ \beta &= 4 + \alpha \end{aligned}$$

Where n is the total number of iterations.

A.4.6 Solving the Vorticity Equation

The solution to the vorticity equation is, unfortunately, not quite as straightforward as the simple equation would lead us to believe. The problem is that vorticity by itself is of limited use in working out how to move the fluid. What we need is a method of obtaining the velocity vector field from the vorticity scalar field (continuing our two-dimensional discussion). To do this, we need to introduce the *streamfunction*.

The concept of *streamlines* should be familiar, at least by name if not by exact definition. A streamline can be considered to be the path that a particle

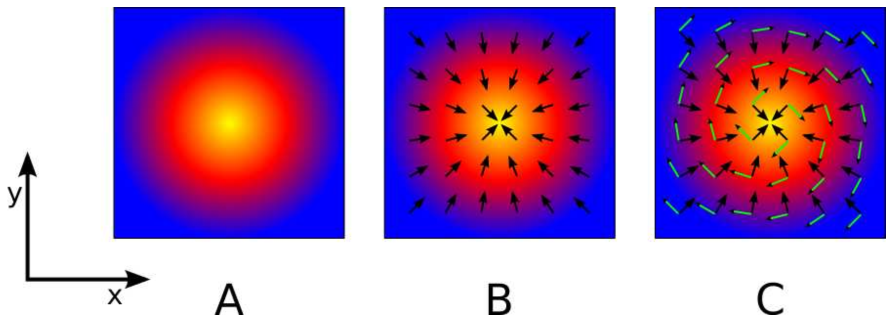


Figure A.5: (a) Visualisation of the streamfunction scalar field. (b) Gradient of the field. (c) Normal of the gradient of the field.

takes through a fluid, and designers of aeroplanes and cars often use wind tunnels to trace streamlines in the flow of air around their products (See Figure A.4). Although quite dramatic when they pass over a well-designed vehicle, it should be noted that streamlines shift and become poorly-defined as turbulence is introduced into the flow.

From their definition, we can see that streamlines directly relate to the velocity field. However, we can also imagine streamlines as following the contours of some scalar field. While it can be difficult to visualise this scalar field, this definition is extremely useful, and the field itself is known as the *streamfunction*. Let us now try to describe this mathematically, sticking to the two-dimensional case.

Consider a streamfunction ψ, visualised in Figure A.5a. Since we are interested in contours of a scalar field, the operator that should immediately come to mind is the gradient. If we take the gradient of the streamfunction, however, it will result in a vector field that points towards the higher value areas of the streamfunction, rather than describing contours (Figure A.5b). What we need to do here is to take the normal of the gradient, by rotating it by 90 degrees (Figure A.5c). This will result in the velocity vector field. We can describe this relationship using a 'normal' function:

$$\mathbf{u} = normal(\nabla\psi) = normal\left(\frac{\partial\psi}{\partial x}, \frac{\partial\psi}{\partial y}\right) = \left(\frac{\partial\psi}{\partial y}, -\frac{\partial\psi}{\partial x}\right)$$

Which, we can see, is the curl of a scalar in two-dimensions. Thus:

$$\mathbf{u} = \nabla \times \psi \tag{A.5}$$

Let us now relate this to vorticity. Recall that:

$$\omega = \nabla \times \mathbf{u}$$

Therefore:

$$\omega = \nabla \times (\nabla \times \psi)$$

Which can be rearranged quite easily if we expand out the curls one at a time:

$$\begin{aligned}
\omega &= \nabla \times \left(\frac{\partial \psi}{\partial x}, -\frac{\partial \psi}{\partial y}\right) \\
\psi &= \frac{\partial}{\partial x}\left(-\frac{\partial \psi}{\partial x}\right) - \frac{\partial}{\partial y}\left(\frac{\partial \psi}{\partial y}\right) \\
\psi &= \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)\psi \\
\psi &= \nabla^2 \psi
\end{aligned}$$

From our discussion on solving the pressure term for the Eulerian Navier-Stokes equations (A.4.5 on page 155), this should be recognisable as a Poisson equation. Thus, using the vorticity we can use the same methods to obtain the streamfunction, from which we can use equation A.5 to calculate the velocity vector field. From this, of course, we can advect our various other properties.

Appendix B

Initial Work

B.1 Sheep

A relatively lightweight solution was required to supply weather effects within a game-like application for an interactive educational tool. Dubbed 'The Meadow', the goal of the project was to allow students to write simple programs which could control the behaviour of a virtual sheep. The sub-goal was to provide a visually-interesting environment, through the use of a variety of modern effects. This work has been presented as [Anderson and McLoughlin 2006a; Anderson and McLoughlin 2006b; Anderson and McLoughlin 2007; McLoughlin and Anderson 2006].

B.1.1 Solution

The solution chosen for the sky was to use two sky-domes, one for the background sky colour and a separate one for the cloud layer. A level of control was offered, with a haze level and a 'weather' variable with which the user (via a GUI control slider) could change the environment from a cloud-less sky through overcast to thunderstorm. Rain effects were added, and an additional light-source for approximating in-cloud lightning.

B.1.2 Background Sky-dome: Sky Colour

Initially, the technique described by Preetham et al. [1999] was implemented on graphics hardware. However, it was realised that while better results were obtained with the algorithm working as a fragment shader, the resulting performance-hit was too great for the target hardware. Conversely, implemented as a vertex shader, although offering acceptable performance, gave a poor quality solution. It was quickly decided that the technique itself was simply inappropriate for the

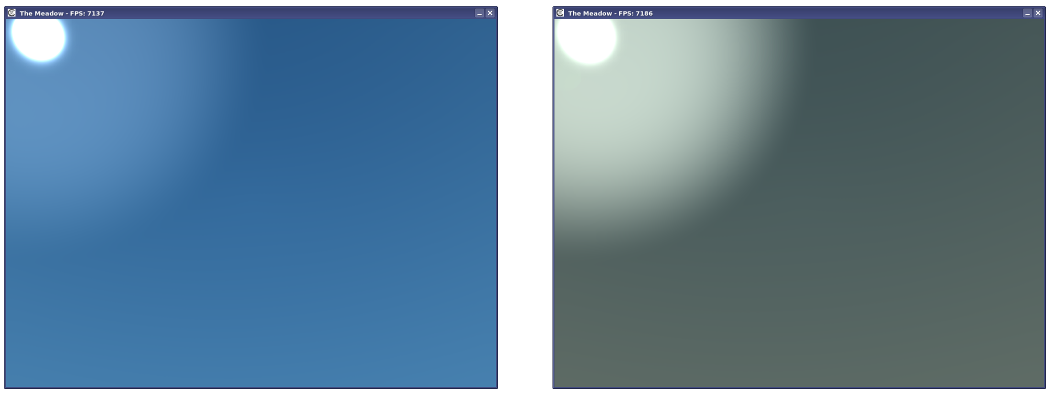


Figure B.1: Left: basic sky-colour within The Meadow; Right: the sky under hazy conditions.

situation, and an empirically-based solution was used instead, based on gradients of colours derived simply from observation. A colour value for the zenith and horizon was determined together with another pair that described the zenith and horizon through an atmosphere of full haze. The position of the sun was not taken into account for the sky gradient, since the simulation assumed a more-or-less constant time of day. Forward scattering effects due to the sun were included as a brightened area around the sun, which was desaturated slightly. (See Figure B.1).

B.1.2.1 Clouds

The solution here was similar to the techniques used by Elias and Pallister [Elias 1998; Pallister 2001], where noise textures were applied to primitive geometry. In this system two octaves of detail were used, as specified by two groups of textures. Each of these groups consisted of a source and destination, which were blended together in a time-varying manner. When the contribution of one or the other was zero, it was swapped with another from a small library of cloud textures. Initially Perlin noise was used for the textures, but it was felt that visually better results were achieved with images derived from photographs, which were loaded in at program initialisation.

The texture-coordinates of the two octaves were determined by a vertex-program, which used projective-texturing based on the sky-dome geometry and a y-coordinate division to emulate a layer that seemed to follow the curvature of the earth. The coordinates were offset with time, at different rates for the two octaves, within the projection transformation to emulate wind and cloud evolution.

The shading of the cloud layer was performed in a fragment shader, and consisted of the following operations:

- Determining the basic cloud sample, constructed from the groups and octaves of cloud textures;
- The application of an exponential-type function, to adjust the cloud coverage based on the current weather value, resulting in a cloud depth value;
- The blending of this sample to a uniform value, based on proximity to the horizon, to emulate distant indistinct clouds;
- The primary shading operation, which applied a set of colours based on the cloud depth, haze level and weather variable;
- The brightening of thin cloud close to the sun;
- The addition of a 'spectral-dispersion' texture, on thin cloud close to the sun, to emulate the optical effect;
- The addition of lighting due to a single lightning source;
- Performing the final tone-mapping procedure, common to all shaders in the system, to apply the exposure operator to bring the high-dynamic-range result into a displayable format;
- Alpha blending was performed on the result, based on the cloud-depth, to remove areas of the cloud layer that did not contain clouds.

Cloud shadows on the ground were not simulated explicitly, but a global value for shadow level was determined (based on the weather value) to approximate total cloud-cover. However, a highly framerate-expensive effect was included that emulated volumetric shafts of light (and cloud shadows) in hazy conditions, using a solution that was inspired by that used by Dobashi [2000]. A series of additional sky-domes were created, each of which being scaled in the y-dimension. The texture-coordinates for these were then determined using the same projective-texturing method as for the cloud layer. The fragment shader then followed the same initial operations as the cloud-layer shader, but performed a different shading routine that resulted in the appearance of volumetric shadows.

Results of the system in various states can be seen in Figure B.2.

B.1.2.2 Weather Effects: Rain

The extreme-end of the weather variable enabled various rain effects. An initial solution was the use of rapidly-scrolling textures, set in front of the camera.

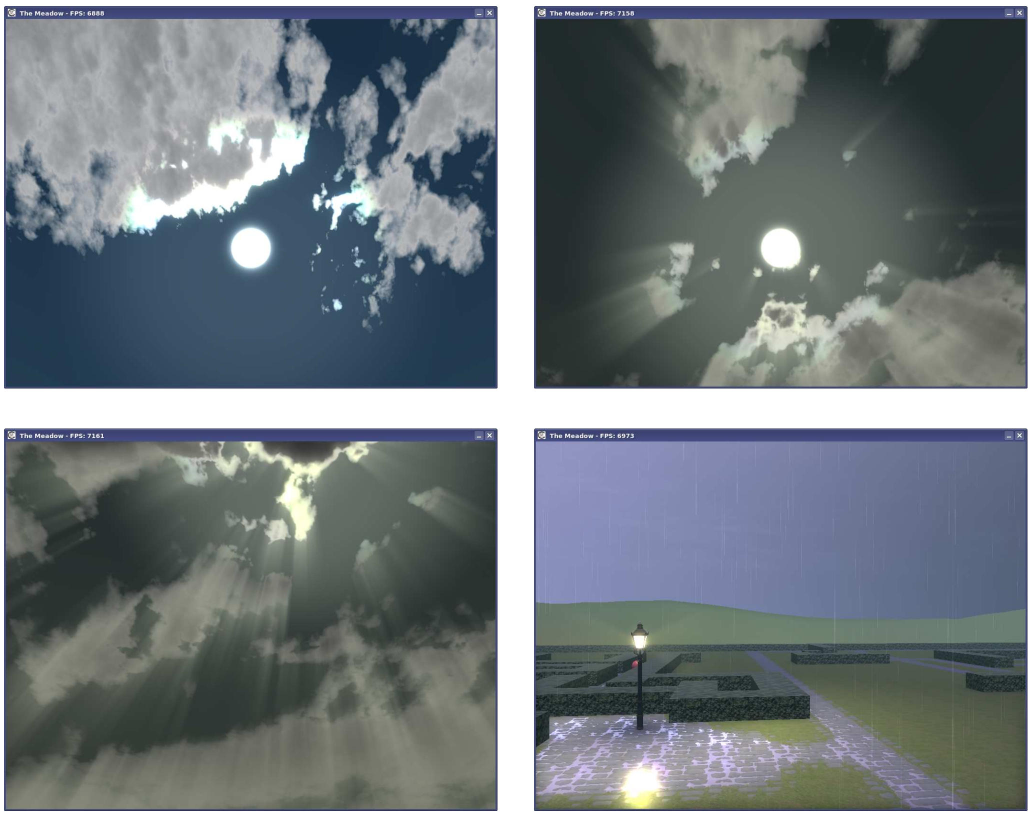


Figure B.2: The dynamic cloud and weather-system within The Meadow.

However, this suffered from a variety of issues, and the illusion was especially poor if the user aimed the camera upwards. This could have been solved by applying the textures to a simple geometric shape, such as used by Wang [2005]. However, it was decided that a simple particle system would be the best approach, with rain drop textures applied to camera-aligned sprites. In addition, locations where the particles impacted with the ground (or, theoretically, other objects — although this was never implemented) could be used to spawn more sprites that played a simple splash animation, as inspired by Tatarchuk and Isidoro [2006].

Further, one of the ground types was a cobble-stone material, and the shaders for this were modified to appear as if water built up in the gaps between the stones. Reflections in this water were made possible by cloning all geometry so that it also existed, mirrored, below ground-level. Alpha blending was then used to show this otherwise hidden geometry.

B.2 Initial Cloud Simulation

The first version of the cloud simulation prototype used a smaller simulation domain of 10km cubed, with different visualisation methods. This version predated NVIDIA's CUDA, which is a generic computing platform allowing C-like programmes to be written for GPUs and which was used in the final prototype. Instead, OpenGL was used and general purpose GPU (GPGPU) techniques were employed to perform the fluid and cloud dynamics simulation, with a simple ray casting method for the rendering. All operations were performed in fragment shaders, written in Cg, and the system ran at interactive rates with a grid resolution of 128 cubed. Visual results are shown in Figure B.3.

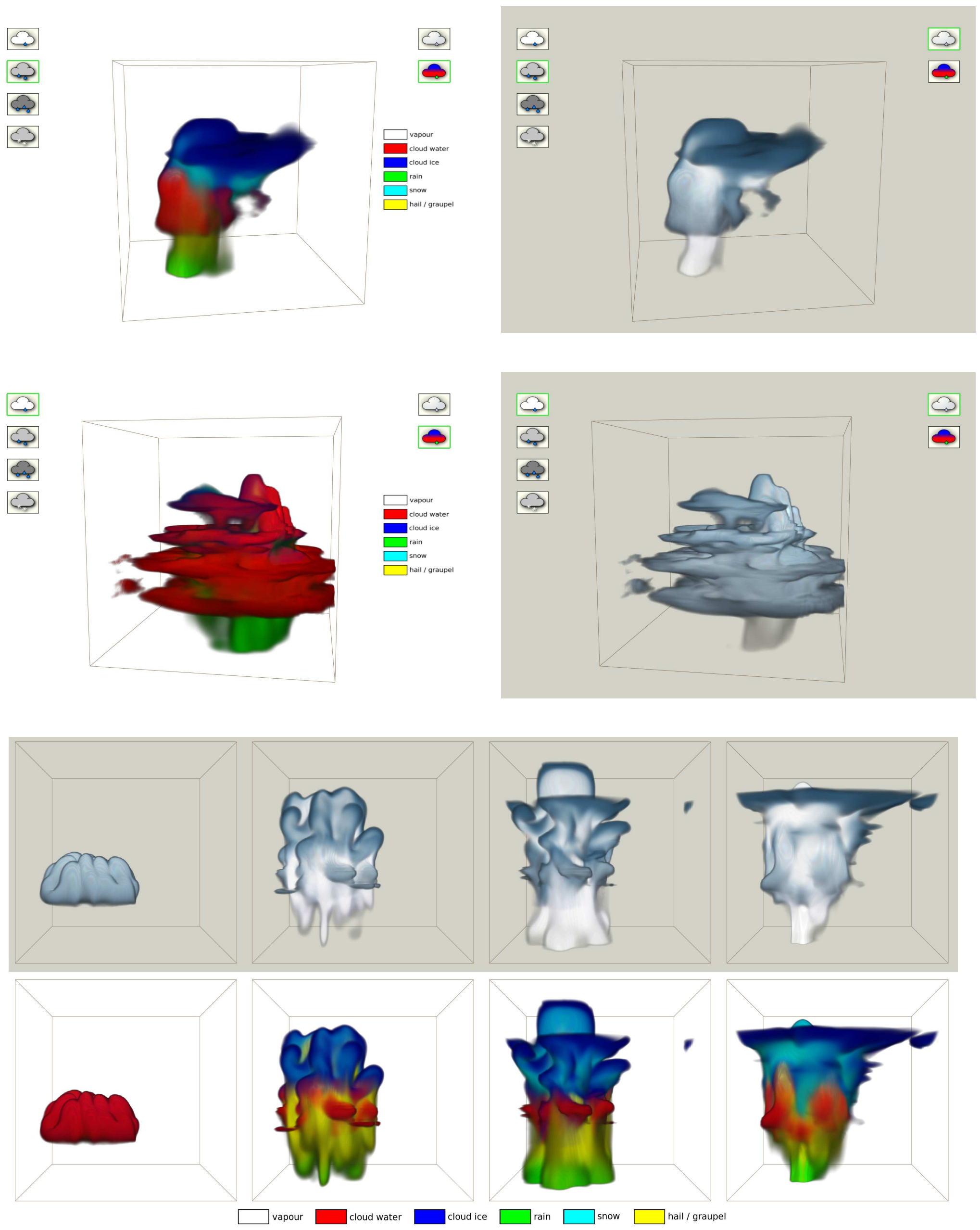


Figure B.3: Visual results from the first 3D prototype system. The water model is essentially the same as the final version. One visualisation technique shows the different water categories, the other was developed with input from the artist Peter Hardie. Top: a small cumulonimbus; middle: a layered formation, achieved by manipulating the temperature profile; bottom: development of a large cumulonimbus — notice how all water categories are present.