Research Proposal

Jake Morey

BSc(Hons) Computer Games Technology
University of Abertay Dundee
School of Arts, Media and Computer Games
May 2014

Abstract

This project will look at simulating and rendering clouds in real time focusing on if it is possible to generate a particular amount of rain depending on the size of the clouds. To show this an application will be created using C++ and DirectX 11 to create and render the scene. While CUDA will be used to simulate the cloud growth on the GPU. The scene will consist of an uneven terrain and a cloud layer all shown in 3D.

This application will then be evaluated using two methods. The first, a visual comparison at different time steps showing the growth of clouds, the rain being created, and the final deformation of the clouds. The second method of evaluation will be to test the efficiency of the model and to see if it could work effectively in an application such as a game. The expected outcome is for it to be possible to generate rain proportional to the size of the cloud. The main focus will be getting it to run in real time and look visually realistic.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Weather:

Since the early 1980's a major part of games have been there use of weather. The weather used in games generally have one of two different uses. The first use of weather in games is using the weather to affect gameplay such as Ouranos! (1980) which uses it as the main game mechanic. More recently Dear Ester (2012) used the weather for the second reason which is as a way to add to the immersion the player feels when playing the game. Figure 1 contains screenshots from both games and showcases how far the weather has come in games. There are some games that utilise both types of weather in games, including Microsoft Flight Simulator X (2003).

Figure 1: Left: Ouranos! (1980), Right: Dear Ester (2012)

Looking closer at the types of weather used in games the use of clouds and rain stand out as the most recurring and notable aspects of weather as well as the most versatile examples of it use in games. For example rain play different roles in two games Heavy Rain (2010) and rain (2013), but with it the games would not be as compelling as the currently are. Figure 2 show screenshots from both games rain (2013) shows how rain is used to affect gameplay by allowing the player only to see the character in the rain, while Heavy Rain (2010) shows how the use of rain adds a depth that increase the film noir feel of the game.

When creating these games a number of different techniques were used to create and render the weather Ouranos! (1980) uses ASCII because of the limitations

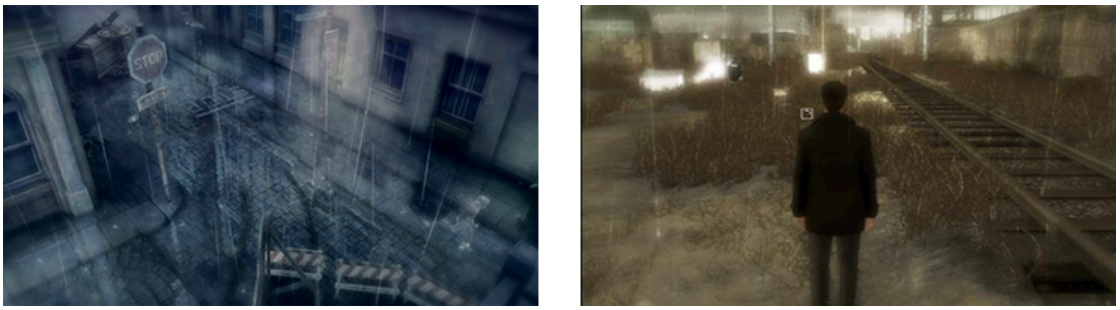

Figure 2: Left: rain (2013), Right: Heavy Rain (2010)

at the time. Games such as Super Mario Bros (1985) use 2D sprites to render clouds on to the background and games like Tomb Raider (2013) uses 3D scripted clouds. 2D games are also more likely to use 2D scrolling rain texture while 3D games are more likely to use a particle system to generate the rain. Some games can use location data to simulate the correct weather at a given location such as NCAA Football 14 (2013). These games usually have a backup dynamic weather system which will be used if the game can't connect to the internet to collect data. Most games created use artistic representation of clouds instead of creating them in real time using equations. Fluid dynamic equations can be used to represent the movement of any object made up of liquid or gas. Basic clouds can be modelled by fluid dynamic equations, more advance clouds need extra equations for water continuity, thermodynamics, and buoyancy. Using equations to generate the clouds will mean that the cloud will behave more like a real cloud than an artist's interpretation of a cloud. When creating the cloud using these equations rain generation can be based proportionally on cloud size or by adding an extra equation to water continuity equations so rainfall is included.

1.2 Research Question:

The project aims to create realistically moving and looking clouds in real time using fluid dynamic equations. Another aim is to create rain in locations and amount that relates to the clouds created. The last aim is to compute the equations in the GPU so that the equations can be computed efficiently. These aims lead to the

research question:

How can the amount of Precipitation generated in a game be related to realistic simulated clouds generated in real-time using fluid dynamic equations?

Answering the research question results in a number of objectives the need to be completed including using a number of equations mainly fluid dynamic equations in 3D space to generate and move clouds. To use these clouds to generate the rain in a 3D scene. The fluid dynamic equations will need to be optimized to run as smoothly as possible in the application whilst look as realistic as possible. The final application will be analysed visually and numerically to test the clouds grow and move overtime as well as generating rain in the most efficient way.

2 Literature

2.1 Background:

Weather plays a huge impact to the how a game feels, as Barton (2008) wrote about how "It was a dark and stormy night" not only sets the time and weather of the scene but also sets the tone. Wang (2004) agrees by saying that one of the most fascinating parts of a scene could be the clouds. An example of this is the game Tomb Raider (2013) which at numerous points in the game the user can see a vast sky full of clouds.

2.2 Cloud Generation:

There are a number of different methods for generating clouds from cellular automaton, to fluid dynamic equations, to importing 3D objects.

2.2.1 Artist Created:

This technique works not by creating the clouds at run time but instead creates the clouds as models and loads them into the game when needed. Wang (2004)

describes a version of this which allows artists to create boxes in 3DS Max in which a plug-in will then generate clouds inside the box. She explained how this method was used when creating Microsoft Flight Simulator 2004: A Century of Flight (2003). A similar method can be used in the CryEngine 3 SDK (2013) which allows the user to alter the properties of the clouds created in real-time within the editor.

2.2.2 Cellular Automaton (CA):

A Cellular Automaton can be described as a regular shaped structure which consists of identical cells that are computed synchronously depending on the state of the cell and its neighbours (Dantchev, 2011). Dobashi *et al.* (2000) used a cellular automaton model when generating the clouds which involved giving each cell a number of boolean states that, when coupled with the rules generated clouds.

This method was extended by Miyazaki *et al.* (2001) who used the Coupled Map Lattice (CML) method which is described as "an extension of cellular automaton, and the simulation space is subdivided into lattices". Miyazaki *et al.* (2001) also goes onto explain that CML model differs from the CA model by using continuous values instead of discreet values. This CML model uses very simple equations for viscosity and pressure effects, advection, diffusion of water vapour, thermal diffusion and buoyancy, and the transition from vapour to water.

2.2.3 Fluid Dynamics:

As clouds can be described as an incompressible fluid it can be simulated via the fluid dynamic equations. The Navier-Stokes Equations are used for a "fluid that conserves both mass and momentum." (Stam, 1999). In the Navier-Stokes equation ρ is the density, $\mathbf{f}$ represents all external forces and ν is the kinematic viscosity of the fluid. The velocity and pressure are defined as $\mathbf{u}$ and p respectively. The

second equation is the continuity equation which means the fluid is incompressible.

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

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

The Navier-Stokes equations (2.1) and (2.2) can be simplified to Euler's Equation because "the effects of viscosity are negligible in gases" (Fedkiw, Stam and Jensen, 2001). This makes the equations for generating the clouds less computationally heavy and can be shown in equation (2.3) which has no $\nu\nabla^2\mathbf{u}$. The continuity equation has not changed and can be seen from (2.4) being the same as (2.2).

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

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

Fedkiw, Stam and Jensen (2001), Harris *et al.* (2003), and Overby, Melek and Keyser (2002) all used work created by Stam (1999) on stable fluid simulations. Overby, Melek and Keyser (2002) used the actual solver created by Stam (1999) in the application to solve the fluid dynamic part of creating clouds. Whereas Fedkiw, Stam and Jensen (2001) and Harris *et al.* (2003) used the theory in the creation of the smoke and clouds respectively. Even though all three used the same start for simulating cloud generation they have different methods for assigning values to the other equations needed.

The Fedkiw, Stam and Jensen (2001) model uses a Poisson equation to compute the pressure of the system and two scalar functions for advecting the temperature and density. This model also uses a function built up of the temperature, ambient temperature, density, and two other positive constants to create a buoyancy effect. The model also simulates velocity fields, which are dampened out on the coarse grid, by finding where the feature should be and then creating a realistic turbulent effect.

Overby, Melek and Keyser (2002) computes the local temperature based upon the heat energy and the pressure. The pressure is calculated from ground level to the tropopause by an exponentially decreasing value (Overby, Melek and Keyser, 2002). The buoyancy in this model is created using the local temperature, the surrounding temperature and a buoyancy scalar. Relative humidity is calculated based upon current water vapour and saturated water vapour. Water condensation is then calculated based upon relative humidity, hygroscopic nuclei, and a condensation constants. The final equation to be calculated is the latent heat which is calculated by the water condensation and a constant.

The Harris *et al.* (2003) model uses equations for water continuity, thermodynamics, buoyancy, and a Poisson equation for fluid flow. This model also creates velocity fields using the same process as Fedkiw, Stam and Jensen (2001). This model uses more complicated equations than the previous two models to more accurately simulate the creation of clouds. For example this model uses gravity, the mass mixing ratio of hydrometeors and virtual potential temperatures, whereas the previous models use scalars or other constants with the temperature to create the buoyancy force.

2.3 Cloud Rendering:

Due to the nature of clouds when light passes through them it becomes scattered. The majority of the models looked at use two different techniques to accomplish this effect single scattering and multiple scattering. These model may render clouds using these scattering techniques directly or may use scattering inside other rendering processes such as photon mapping. A number of these models also use billboards or imposters when rendering the clouds as this saves on computation. This section will look at single scattering, multiple scattering, and photon mapping.

2.3.1 Single Scattering:

Harris and Lastra (2001) describe single scattering as a model that simulates scattering in a single direction that is usually the direction leading to the point of view. There are debates whether or not this type of rendering is detailed enough for rendering clouds. Miyazaki *et al.* (2001) states the main topic of his model is the cloud shapes so using single scattering is enough to check the shape of the cloud. Bohren (1987) describes single scattering as insufficient when describing common observations.

2.3.2 Multiple scattering:

"Multiple scattering models are more physically accurate, but must account for scattering in all directions ... and therefore are much more complicated and expensive to evaluate" (Harris and Lastra, 2001). Harris *et al.* (2003) uses a version of multiple scattering which is called multiple forward scattering, this differs from the original by instead of calculating scattering in all directions it calculates scattering in the forward direction only. This means the algorithm is less computationally heavy. Fedkiw, Stam and Jensen (2001) describe the multiple scattering of light as necessary for objects made from water vapour, which clouds are.

2.3.3 Photon Mapping:

"Photon mapping is a variation of pure Monte Carlo ray tracing in which photons (particles of radiant energy) are traced through a scene" (Jensen (1996), cited in Harris (2003)). Harris (2003) describes the process of photon mapping as storing position, incoming direction, and radiance of each photon landing on a nonspecular surface that has been traced from the light source. Fedkiw, Stam and Jensen (2001) use photon mapping when rendering smoke and describe the process as a two pass algorithm, one where a volume photon map is built and the second as a rendering pass using a forward ray marching algorithm.

2.4 Rain Rendering:

"Rain is an extremely complex natural atmospheric phenomenon" (Puig-Centelles, Ripolles and Chover, 2009). Puig-Centelles, Ripolles and Chover (2009) describe two main techniques for rendering rain to a scene scrolling textures where a texture the size of the screen scrolls in the direction of the rain, and a particle system where each rain drop is represented as a particle in the system. Tariq (2007) writes "animating rain using a particle system is more useful for realistic looking rain with lots of behaviour (like changing wind)." Whereas Puig-Centelles, Ripolles and Chover (2009) sates texture scrolling "is faster than particle systems, but it does not allow interaction between rain and the environment."

3 Methodology:

3.1 Practical Methodology:

In answering the research question an application will be created showing the generation of clouds in real time as well as generating different amounts of rain falling from these clouds when they start to dissipate. This application will be written in C++ using Visual Studio as an IDE. It will also use Nvidia's CUDA general-purpose GPU computing language to simulate the growth and dissipation of the clouds in real time. The DirectX 11 API will be used to render, this data created from the CUDA processes, to the screen.

In the application the scene will consist of two things an uneven terrain and a layer for developing clouds. Much like the model used by Dobashi *et al.* (2000) which is shown below in Figure 3.1. Snapshots from Harris *et al.* (2003) and Miyazaki *et al.* (2001) are both shown in the appendix as Figure 6.1 and Figure 6.2 respectively. The clouds will be generated using fluid dynamic equations and rendered using multi-scattering techniques.

Before creating this 3D model a simple two dimensional model will be created to test how detailed the other equations that are needed in the model should be.

For example should these equations be as complex as the Harris *et al.* (2003) model or simpler like the Fedkiw, Stam and Jensen (2001) model.

3.2 Evaluation Methodology:

To evaluate the application there will be two methods used both based upon methods found in the literature review. The first is a visual test at time steps during the running of the program. This will hopefully showcase the formation and deformation of clouds and the generation of rain. Figure 3.1 shows Dobashi *et al.* (2000) results in the manner describe here. In the appendix Figure 6.2 to Figure 6.4 show the results from other models. This method has been chosen because it shows of the purpose of the project which is to have visually realistic clouds simulated at real time with the creation of rain.

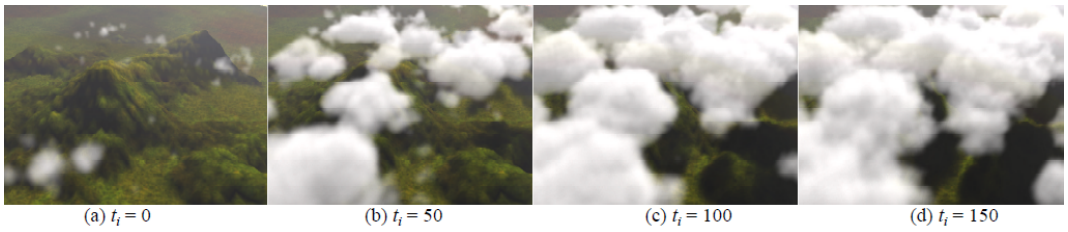

Figure 3: Dobashi *et al.* (2000)

The other method that will be used is to test how much resources are being used by the application. Harris and Lastra (2001), and Elek *et al.* (2012) both used this method when evaluating their models. This form of evaluation will be used to test how efficient the application is when running in real time as well as concerns like the size of grid that can be used. This can be accomplished using NSight Visual Studio Edition for debugging CUDA and profiling the application.

3.3 Risk Analysis:

There aren't that many risks when developing the application except for the fact that the simulation of clouds might not be able to be rendered in real time. Another

risk would be not finding the correct moment in the clouds life cycle in which to start and finish generating the rain. To combat this a 2D simulation will be created first which will be used to test different moments of the clouds diffusion for the right moment to start generating rain.

3.4 Project Plan:

To start the project a basic two dimensional application will be created that will test the complexity needed for the other functions being used in this model such as the buoyancy equation. This model will also help when figuring out the moment to generate rain. After this application is created a basic 3D application will be created which will consist of an uneven terrain and a camera. The next part of the project to be undertaken will be turning the 2D simulation code into 3D simulation code. After there is code for creating a 3D simulation of the clouds the next step will be to render these clouds in 3D. Next generating the rain will be added to the application. This step will allow for the created and destruction of the rain at the right moments of the clouds life. There are two more steps related to the application in the project plan. The first being optimisation and the second being evaluation. There will be some overlap between these two stages as the optimisation could lead to an increase in application efficiency and will be part of the evaluation methodology.

There are two other stage that will be carried out whilst the application is being created and tested. These are writing the draft dissertation and the final dissertation. The reason these will be written during the creation of the application and not after testing is to give enough time to write the dissertation to a high standard.

This timetable can be seen in Gantt Chart form in the Appendix as Figure 6.5.

4 Results:

5 Discussion:

6 Conclusions:

References

Barton, M. 2008. How's the weather: Simulating weather in virtual environments 8. Available from: http://gamestudies.org/0801/articles/barton. 2.1

Bohren, C. F. 1987. Multiple scattering of light and some of its observable consequences. *Am.J.Phys* 55(6). pp. 524–533. 2.3.1

Crytek 3 SDK. 2013. *CryEngine* [software] Version 3. Available from: http://mycryengine.com/. 2.2.1

Dantchev, S. 2011. Dynamic neighbourhood cellular automata. *Computer journal* 54(1). pp. 26–30. 2.2.2

Dear Esther. 2012. [Online]. PC. The Chinese Room. (document), 1.1, 1

Dobashi, Y. et al. 2000. A simple, efficient method for realistic animation of clouds. ACM Press/Addison-Wesley Publishing Co. pp. 19–28. (document), 2.2.2, 3.1, 3.2, 3

Elek, O. et al. 2012. Interactive cloud rendering using temporally coherent photon mapping. *COMPUTERS and GRAPHICS-UK* 36(8). pp. 1109–1118. 3.2

Fedkiw, R., Stam, J., and Jensen, H. 2001. Visual simulation of smoke. ACM. pp. 15–22. (document), 2.2.3, 2.2.3, 2.3.2, 2.3.3, 3.1, 6

Harris, M. et al. 2003. Simulation of cloud dynamics on graphics hardware. Eurographics Association. pp. 92–101. (document), 2.2.3, 2.3.2, 3.1, 4, 7

Harris, M. J. 2003. *Real-time cloud simulation and rendering.*. [Online]. 2.3.3

Harris, M. and Lastra, A. 2001. Real-time cloud rendering. *Computer Graphics Forum* 20(3). pp. C76–C76. 2.3.1, 2.3.2, 3.2

Heavy Rain. 2010. [Disk]. PlayStation 3. Sony Computer Entertainment. (document), 1.1, 2

Jensen, H. W. 1996. *Global illumination using photon maps*. Springer. pp. 21–30. Rendering Techniques 96. 2.3.3

Microsoft Flight Simulator 2004: A Century of Flight. 2003. [Disk]. PC. Microsoft. 1.1, 2.2.1

Miyazaki, R. et al. 2001. A method for modeling clouds based on atmospheric fluid dynamics. In: *Computer Graphics and Applications, 2001. Proceedings. Ninth Pacific Conference on*. IEEE. pp. 363–372. (document), 2.2.2, 2.3.1, 3.1, 5

NCAA Football 14. 2013. [Disk]. PlayStation 3. EA Sports. 1.1

Ouranos! 1980. [Cassest]. Commodore PET. The Code Works. (document), 1.1, 1, 1.1

Overby, D., Melek, Z., and Keyser, J. 2002. Interactive physically-based cloud simulation. In: *Computer Graphics and Applications, 2002. Proceedings. 10th Pacific Conference on*. IEEE. pp. 469–470. 2.2.3

Puig-Centelles, A., Ripolles, O., and Chover, M. 2009. Creation and control of rain in virtual environments. *VISUAL COMPUTER* 25(11). pp. 1037–1052. 2.4

rain. 2013. [Online]. PlayStation 3. Sony Computer Entertainment. (document), 1.1, 2

Stam, J. 1999. Stable fluids. ACM Press/Addison-Wesley Publishing Co. pp. 121–128. 2.2.3, 2.2.3

Super Mario Bros. 1985. [Cartridge]. Super Nintendo Entertainment System. Nintendo. 1.1

Tariq, S. 2007. Rain. Tech. rep.. NVIDIA. 2.4

Tomb Raider. 2013. [Online]. PC. Square Enix. 1.1, 2.1

Wang, N. 2004. *Let there be clouds! fast, realistic cloud-rendering in microsoft flight simulator 2004: A century of flight.* [Online]. Available from: http://www.gamasutra.com/view/feature/130434/let_there_be_clouds_fast_.php. 2.1, 2.2.1

Appendix:

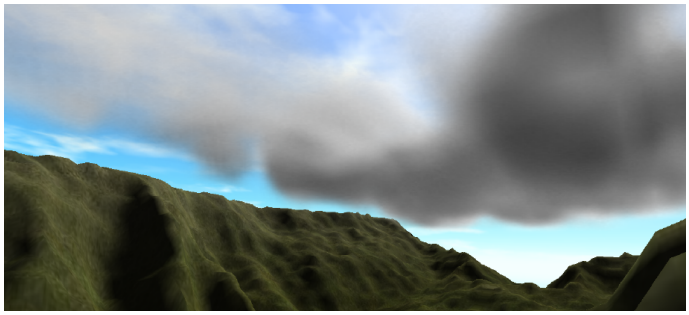

Figure 4: Harris *et al.* (2003)

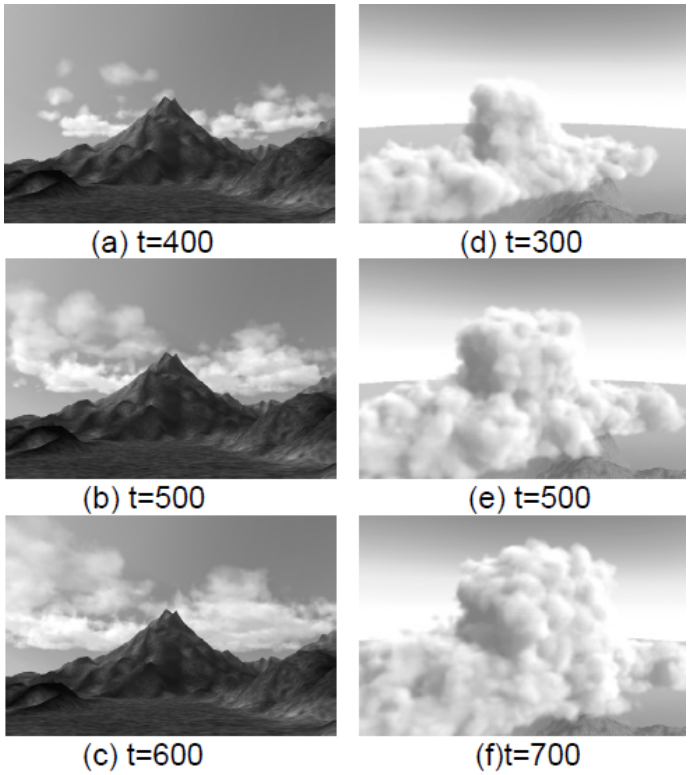

Figure 5: Miyazaki *et al.* (2001)

Figure 6: Fedkiw, Stam and Jensen (2001)

Figure 7: Harris *et al.* (2003)

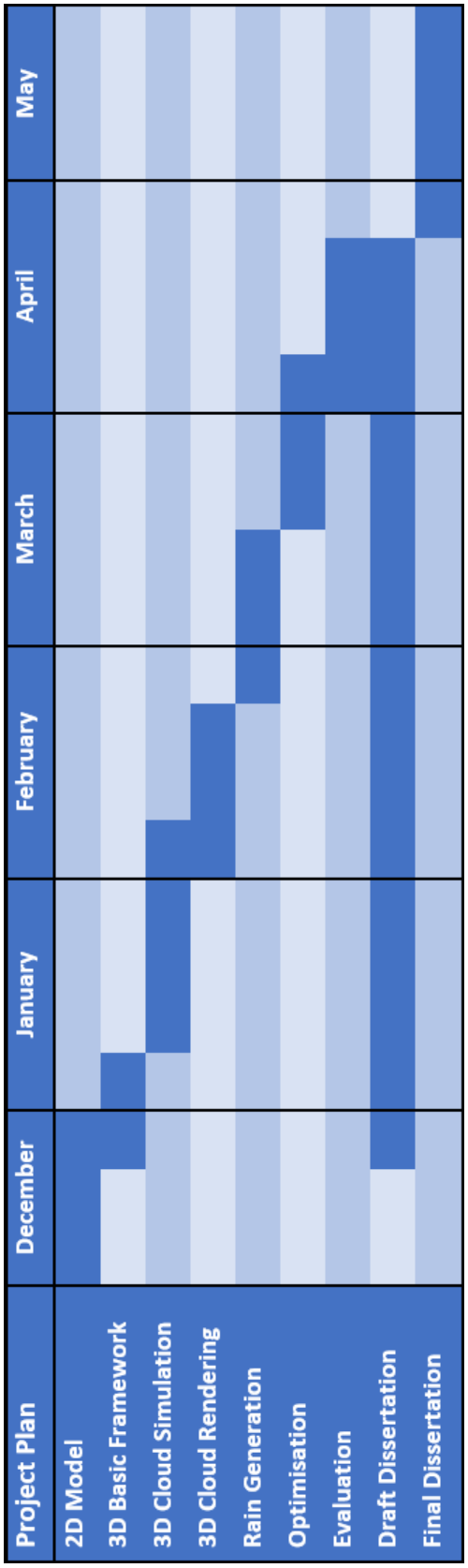

Figure 8: Project Plan