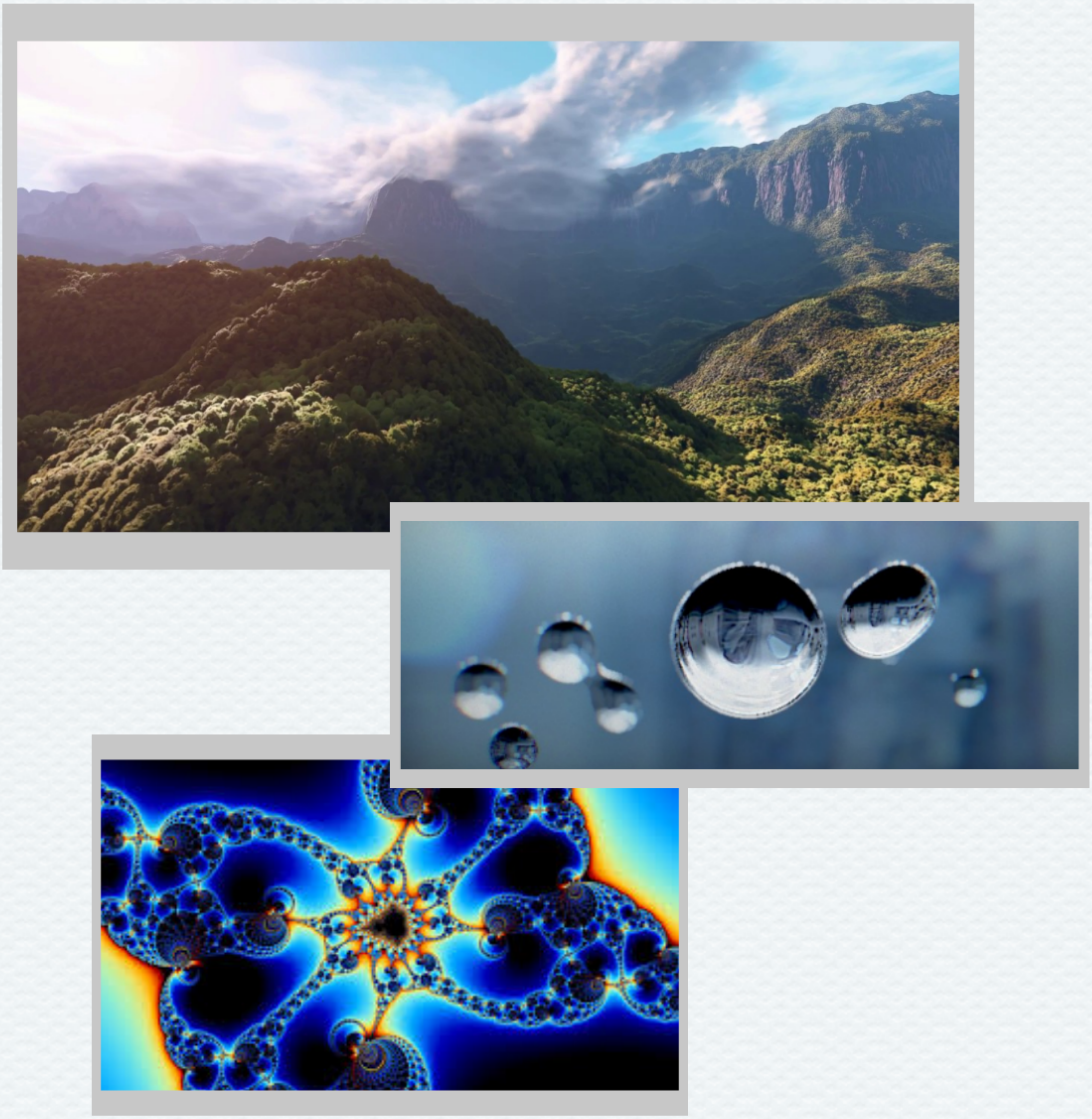
The Design and Implementation of an Advanced Graphics Screensaver Suite

By James Balajan



Contents

1 Project Proposal **1**
1.1 Executive Summary . . . 1

2 Project Plan **2**
2.1 Defining and Understanding the Problem . . . 2
2.2 Design . . . 2
2.3 Implementation . . . 3
2.4 Testing . . . 3

3 Feasibility Study **4**
3.1 Executive Summary . . . 4
3.2 Description of Products and Services . . . 4
3.3 Technology Considerations . . . 5
3.3.1 Hardware Considerations . . . 5
3.3.2 Software Considerations . . . 5
3.4 Product/Service Marketplace . . . 5
3.5 Social and Ethical Considerations . . . 6
3.5.1 Legal Considerations . . . 6
3.5.2 Ergonomics . . . 6
3.5.3 Inclusivity . . . 6
3.6 Organization and Staffing . . . 7
3.7 Schedule . . . 7
3.8 Financial Projections . . . 7
3.9 Findings and Recommendations . . . 8

4 Design **8**
4.1 System Modeling Diagrams . . . 8
4.2 Modularity and Design Patterns . . . 9
4.2.1 Clouds and Mountains Screensaver . . . 9
4.2.2 Mandelbrot Screensaver . . . 10
4.2.3 Metaballs Screensaver . . . 10
4.3 Influence of Survey Responses on Design and Direction . . . 11
4.4 Configuration Windows . . . 12

5 Implementation **13**
5.1 Mandelbrot Screensaver . . . 13
5.1.1 Point Mapping . . . 13
5.1.2 Colouring . . . 14
5.2 Metaballs Screensaver . . . 14
5.2.1 Raymarched Scene . . . 14
5.2.2 The SDF Raymarching Algorithm . . . 15
5.3 Clouds and Mountains Screensaver . . . 16
5.3.1 Atmospheric Scattering . . . 16
5.3.2 Volumetric Raymarching . . . 17
5.3.3 Procedural Terrain . . . 19

5.3.4 God Rays . . . 19

6 Testing **20**

7 Evaluation **22**

Bibliography **24**

Glossary **25**

8 Appendix **27**
8.1 Designs . . . 27
8.1.1 Clouds Screensaver Context Diagram . . . 27
8.1.2 Clouds Screensaver Data Flow Diagram . . . 28
8.1.3 Clouds Screensaver Structure Chart . . . 29
8.1.4 Clouds Screensaver Storyboard . . . 30
8.1.5 Mandelbrot Screensaver Context Diagram . . . 31
8.1.6 Mandelbrot Screensaver Data Flow Diagram . . . 32
8.1.7 Mandelbrot Screensaver Structure Chart . . . 33
8.1.8 Mandelbrot Screensaver Storyboard . . . 34
8.1.9 Metaballs Screensaver Context Diagram . . . 35
8.1.10 Metaballs Screensaver Data Flow Diagram . . . 36
8.1.11 Metaballs Screensaver Structure Chart . . . 37
8.1.12 Metaballs Screensaver Storyboard . . . 38
8.1.13 Configuration Window Screen Design . . . 39
8.1.14 Clouds and Mountains Screensaver Screen Design . . . 40
8.1.15 Mandelbrot Screensaver Screen Design . . . 41
8.1.16 Metaballs Screensaver Screen Design . . . 42
8.1.17 Data Dictionary . . . 43
8.2 Survey Responses . . . 65
8.2.1 Julia Balajan - Survey Response . . . 65
8.2.2 Ms Saki - Survey Response . . . 67
8.2.3 Arjun Malik - Survey Response 1 . . . 69
8.2.4 Arjun Malik - Survey Response 2 . . . 70
8.2.5 Samuel Stacy - Survey Response 1 . . . 72
8.2.6 Samuel Stacy - Survey Response 2 . . . 73
8.2.7 Julian Peen - Survey Response 1 . . . 75
8.2.8 Julian Peen - Survey Response 2 . . . 76

List of Figures

1.1 Inspiration for Screensaver. Computer generated Rainforest by Inigo Quilez 1
1.2 Graphics techniques that may be used 1

2.1 High-level diagram summarising project plan 3

3.1 Windows 98 Maze 4
3.2 Windows XP Pipes 4
3.3 From left to right example images of: Terrain with Volumetric Clouds, Mandelbrot Fractal and Metaballs. 5
3.4 Examples of mandelbrot and metaballs products 6
3.5 Gantt chart 7

4.1 From left to right: Mandelbrot Screensaver Context Diagram, Clouds Screensaver DFD. For full resolution versions refer to appendix 8
4.2 From left to right: Metaballs Screensaver Structure Chart, Clouds Screensaver Storyboard. For full resolution versions refer to appendix 9
4.3 Flyweight Forest 10
4.4 Configuration Window Message Box 12

5.1 Mandelbrot Equation 13
5.2 Real and Imaginary Component Mapping to Cartesian Plane 13
5.3 Smooth Iteration Count Formula from (Quilez, 2016) 14
5.4 Smooth vs. Discrete Mandelbrot Fractals from (Quilez, 2016) 14
5.5 Ray tracing from (Henrik, 2008) 15
5.6 Sphere tracing illustrated from (Pharr, 2005) 15
5.7 The Phase Function from (Pharr, 2005) 16
5.8 The Out-Scattering Equation from (Pharr, 2005) 17
5.9 The In-Scattering Equation from (Pharr, 2005) 17
5.10 From left to right example images of: Perlin FBM Noise, Worley Noise, Perlin-Worley Noise 17
5.11 From left to right example images of height gradients for: Stratus, Cumulus and Cumulonimbus clouds from (Engel, 2016) 17
5.12 Image illustrating the 4 steps of volumetric raymarching from (Hofmann, 2011) 18
5.13 Beer-Lambert Law 18
5.14 Beer's-Powder function 19
5.15 From left to right: Typical heightmap, Mesh viewed in RenderDoc Debugger 19
5.16 Equations representing the recurrent determination of the amount of light absorbed from the point of emission to the view point as well as the amount due to light scattering into the path of the view ray from (Nguyen, 2007) . . 19

6.1 Screensaver not rendering 20
6.2 User Windows registry showing expected values 21
6.3 God rays jumping 21
6.4 God ray jittering bug fixed 21

7.1 From left to right: Mandelbrot screensaver previewed in Windows screensaver selection GUI, Mountains screensaver Settings GUI 23

8.1 Clouds Screensaver Context Diagram 27
8.2 Clouds Screensaver Data Flow Diagram 28
8.3 Clouds Screensaver Structure Chart 29
8.4 Clouds Screensaver Storyboard 30
8.5 Mandelbrot Screensaver Context Diagram 31
8.6 Mandelbrot Screensaver Data Flow Diagram 32
8.7 Mandelbrot Screensaver Structure Chart 33
8.8 Mandelbrot Screensaver Storyboard 34
8.9 Metaballs Screensaver Context Diagram 35
8.10 Metaballs Screensaver Data Flow Diagram 36
8.11 Metaballs Screensaver Structure Chart 37
8.12 Metaballs Screensaver Storyboard 38
8.13 Configuration Window Screen Design 39

Chapter 1

Project Proposal

Figure 1.1: *Inspiration for Screensaver. Computer generated Rainforest by Inigo Quilez*

1.1 Executive Summary

Much promise has been shown in the applicability of basic, computer generated graphics for creating screen savers for computers in the past, an example of this being that all of the screensavers for Windows XP were rendered. The ability to create aesthetically pleasing, dynamic graphics has made computer generated graphics find use in this field. For this reason a study will be undertaken to investigate if utilising advanced computer graphics techniques to generate a visually appealing scene, which will act as a screensaver, is possible without being overly burdensome on the computer's hardware. Some proposed techniques to be utilised to make this scene are physically based rendering, atmospheric scattering and volumetric raymarching, though this list of techniques is subject to change throughout the project as the needs and challenges of the project cannot fully be anticipated.

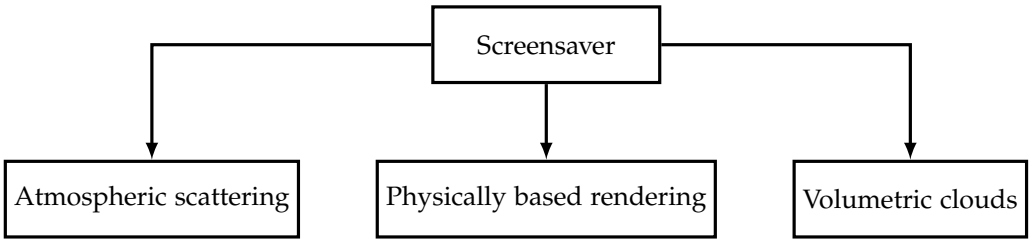


Figure 1.2: *Graphics techniques that may be used*

To manage the project, the structured approach will be utilised and documented in a portfolio. Our clientele would like for us to use VB.NET to create the project, so the project will be programmed utilising it. As a result, challenges may be faced regarding the performance of the screensaver. Since the development of this screensaver will be only undertaken by one developer, the anticipated completion date of the project is June 2019, although this deadline is subject to change based on circumstances regarding the project.

Chapter 2

Project Plan

2.1 Defining and Understanding the Problem

The initial thoughts on what software construction methodology to use leaned towards one which would benefit a small development team, such as the prototyping methodology. However, it is mandated by our project sponsor that a portfolio be developed to document our development process in a structured manner, so a structured approach is the best choice due to its reliance on documentation.

Thus, the project will be split into 5 stages according to the structured approach:

- Planning
- Design
- Building / Implementing
- Testing
- Deployment

These stages are reflected in our Gantt Chart. Please see figure 3.5 on page 7 for the Gantt Chart.

During the design phase, a survey will be conducted, asking customers what features they would like in a screensaver suite. The survey responses will be considered in our designs and implementation. After initial implementation, a survey will be conducted requesting the opinions of our customers on the product. As of such, there will be some elements of the prototyping methodology in our approach as well, iterating over our initial product to improve it and satisfy the expectations of our customers. However, as stated previously, the methodology of development will be primarily structured.

2.2 Design

When developing the designs for the product, the following will be created.

- Context Diagrams.
- Data Flow Diagrams.
- Structure Charts.
- Story board.
- A Data Dictionary.

All besides the data dictionary will be created before the implementation stage. The data dictionary will be created along with the product itself.

2.3 Implementation

The implementation stage will firstly begin with a technical research stage. Advanced graphics techniques involve use of heavy amounts of mathematics, and as of such, research papers must be studied and cited.

After the initial research, implementation will commence. Testing will be done regularly as the project progresses, to ensure each module is functional before moving onto the next one whilst avoiding the accumulation of errors.

2.4 Testing

On completing the implementation, the screensaver suite will be tested on a variety of hardwares, to ensure that it functions correctly on other computer hardwares. This is critical, as graphics shaders may perform differently and possibly unexpectedly based on each graphics card provider's OpenGL implementation.

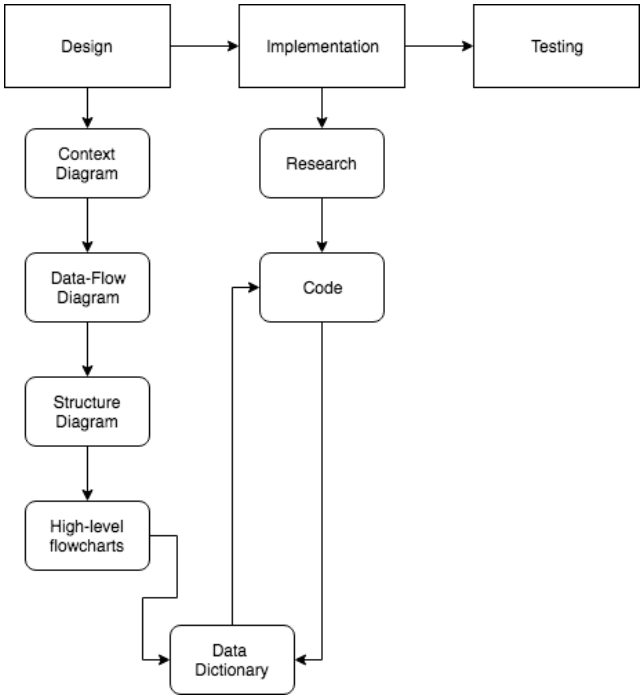


Figure 2.1: *High-level diagram summarising project plan*

Chapter 3

Feasibility Study

3.1 Executive Summary

Screensavers have been a medium of graphics experimentation during the times of early computing. However, advanced graphics techniques have never been applied to the context of screensavers, leaving an excellent vacant marketplace to test cutting edge techniques in the context of screensavers. This document analyzes the feasibility of undertaking a project where advanced graphics techniques are applied to create a screensaver from the lenses of technological feasibility, staffing feasibility and marketplace feasibility.

3.2 Description of Products and Services

Screensavers, during the times of early computing, have been the medium for the showcasing of various graphics techniques. From the Windows 98 era with the notable screensaver maze, bringing rasteurized graphics to the customer in a display of engineering creativity to the classic Windows XP pipes, using random generation to provide the customer with a new and consistently unique scene, graphics techniques have found their place in the world of screensavers. However, these screensavers have often been quite basic, using simple Phong shading or even no shading at all. This makes one wonder why advanced graphics techniques, such as the ones explored by video game companies and animation studios such as Pixar, have not been explored in the avenue of screensaver creation.

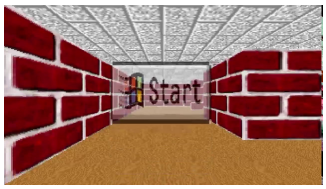


Figure 3.1: *Windows 98 Maze*

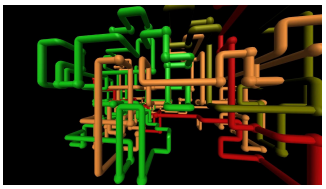


Figure 3.2: *Windows XP Pipes*

This project aims to investigate this avenue, with the aim of creating a suite of 3 screensavers. The first one will investigate the application of volumetric clouds and randomly generated terrain to screensavers. The second will investigate how mathematical fractal geometry may be applied to generating interesting screensavers. The final will investigate the application of the interesting effect of metaballs to screensavers.

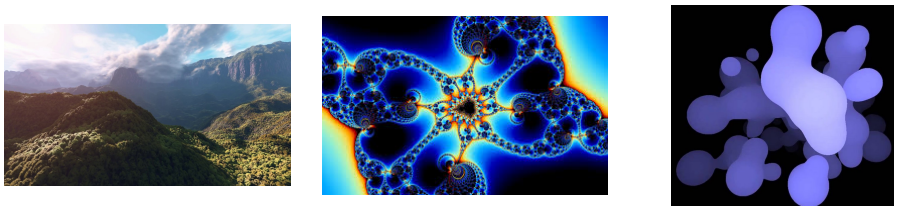


Figure 3.3: *From left to right example images of: Terrain with Volumetric Clouds, Mandelbrot Fractal and Metaballs.*

3.3 Technology Considerations

3.3.1 Hardware Considerations

The personnel possess a single HP Pavilion laptop running Windows 10 OS that uses a Nvidia Geforce GTX 1050 graphics card. Although the graphics card is powerful, it is not out of reach of most customers, making it an excellent benchmark for what our users may use.

3.3.2 Software Considerations

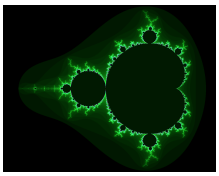
The development will be carried out using the VB.NET programming language through the Visual Studio IDE. VB.NET was chosen as a matter of preference from our project sponsor. Due to this unorthodox choice for a programming language for graphics programming, in which C and C++ are most commonly used due to their performance, extra considerations may need to be taken regarding the overhead of the VB.NET language. Furthermore, as VB.NET is being used to create the screensaver suite, the project is only going to work on the Windows operating system as it requires the .NET framework. However, projects such as .NET core and Mono aim to port the .NET framework to linux and mac, so cross compatibility is not completely out of the question, and can be implemented later if the demand is ever shown.

3.4 Product/Service Marketplace

The marketplace for the utilisation of advanced graphics techniques in the context of screensavers has been fairly limited. No repositories on github were found for screensavers which utilised volumetric clouds, which is not surprising as it is a relatively new graphics technique which is still being researched. No procedural terrain screensavers were found either. This makes the first screensaver idea, of procedurally generated terrain with volumetric clouds, a venture into entirely uncharted territory, as the marketplace is entirely empty.

As for the mandelbrot fractal screensaver, mandelbrot fractals are well documented mathematical structures which produce beautiful results when rendered, however have also, surprisingly not been applied in the context of screensavers. There however exists a large number of mandelbrot fractal explorers generated, as it is seen as an excellent exercise to develop one's programming skills, with one shown in figure 3.4.

Metaballs are also a well documented technique. Metaballs have been applied to a wide range of areas, however, one particular use of them is in simulating the behaviour of water droplets efficiently, without the use of complex and expensive fluid dynamics simulations. Metaballs have not been applied to the context of screensavers, however, various eye candy demos have been created such as the one in figure 3.4.



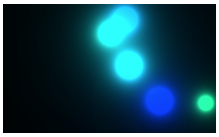


Figure 3.4: *Left to right: Mandelbrot drawn by Rafrex's "Fractal", Metaballs rendered by FlannelHead's "Dynaballs"*

3.5 Social and Ethical Considerations

3.5.1 Legal Considerations

As this project is primarily a non-profit, research experiment, the project will be released using the Open Source MIT License. This license limits our liability and makes warranty not legally enforcable, legally protecting us in the case of a potential misuse of our software. The license permits commerical use, modification, distribution and private use of our not for profit, research project.

3.5.2 Ergonomics

- The screensaver should be designed to be ergonomic and user friendly. This is easily achievable, as screensavers lack the traditional GUIs of many other products, their main purpose being to be eye candy. Options for these screensavers may be programmed to be directly accessible through the Windows control panel, or could be programmed to be configurable through a separate GUI all together.
- The design of the screensaver options dialog will aim to keep the interface simple and intuitive to use, not overwhelming the user with unnecessary options.
- The screensavers will aim to maximise FPS without jeopardising the quality of the screensavers.

3.5.3 Inclusivity

- An English locale will be implemented into the screensaver suite options dialog, as English is one of the most widely spoken languages in the world. Furthermore, a Russian locale will be implemented as it is the 7th most widely spoken language in the world, and the developer has knowledge of the language. As this project is open source, if more locales are desired to be programmed in, open source contributors may add them.
- Data formats that differ internationally, such as dates and currency, are not being handled and hence these factors do not need to be accounted for.
- The screensaver would be free and released under the open source licence, allowing anyone of any economic background to utilise it.
- The scenes rendered in the screensavers will be designed to be universally appealing to all demographics.
- The option for the screensaver to play music may be programmed in at a later date. One option used to cater for people with hearing disabilities in software is to include subtitles. However, this solution is not suitable, as the person would still not be able to experience the music through a simple text prompt which says "Music Plays". As

of such, this issue will be left to the open source community to consider, if a solution is found in the future.

- People with visual impairments must also be catered for. Legal blindness cannot be accounted for, as the product is a purely visual one. However, for people with mild visual impairments, text can be written using a clear serif or sans-serif font with a large font size. Poor color combinations will be avoided like yellow on white, only using colors which contrast well, such as black on white.

- Issues pertaining to cultural inclusivity need not be worried about, as the screensavers aim to have universal beauty, showcasing the remarkable natures of both the natural and mathematical worlds.

3.6 Organization and Staffing

Considerations must be made regarding whether the developers of this project possess the expertise and the technology required to complete the project. The development team consists of one student programmer, indicating that the personnel may lack the expertise required to develop an advanced graphics screensaver. However, the deadline for the end of the project provides enough time for the programmer to research and become knowledgable on these techniques before implementation.

3.7 Schedule

The schedule of this project will closely follow the Gantt Chart. The structure of this gantt chart aims to reflect the standard software development life cycle.

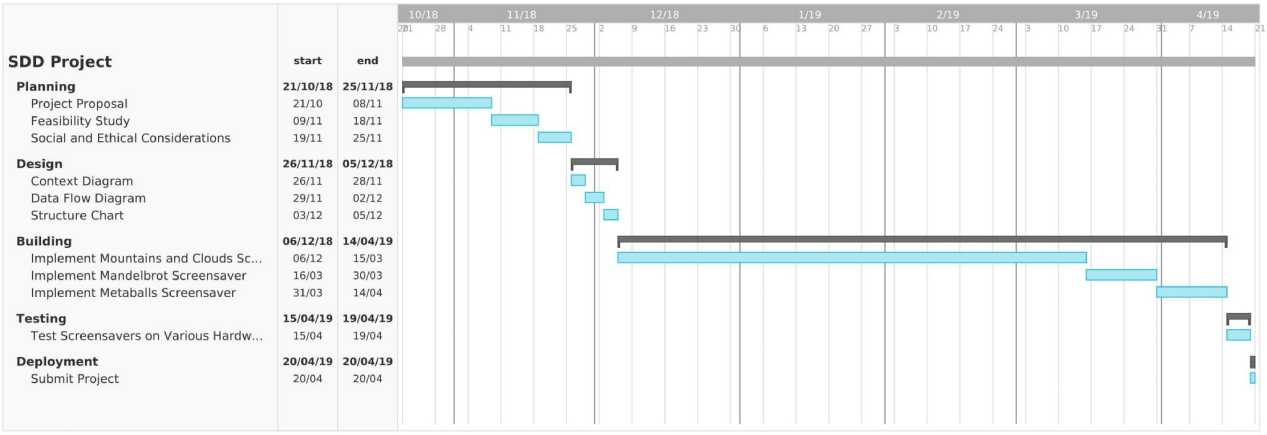


Figure 3.5: *Gantt chart*

3.8 Financial Projections

As the project manager is also the developer and has direct interests in completing this project, the developer demands no pay. No extra software must be purchased, as all tools that will be used are free. The only costs associated with this project are those associated with electricity for recharging the laptop.

3.9 Findings and Recommendations

In summary:

- There is ample time for research, development and refinement.
- Through investigating advanced graphics techniques in screensavers, one can set the groundwork for their future integration into other mediums such as animated movies and video games.
- The cost of development is negligible.

Hence, it is in our best interests that the project proceed.

Chapter 4

Design

4.1 System Modeling Diagrams

As a screensaver product is primarily visual, the first part of the design stage was to create an array of preliminary screen designs, with one for each of the three screensavers in the suite. These screen designs may be found in the appendix according to the list below:

- Clouds and Mountains Screen Design (pg. 40)
- Mandelbrot Screen Design (pg. 41)
- Metaballs Screen Design (pg. 42)

In the design stage, a detailed context diagram, data flow diagram (DFD), structure chart and storyboard were also created for each of the three screensavers.

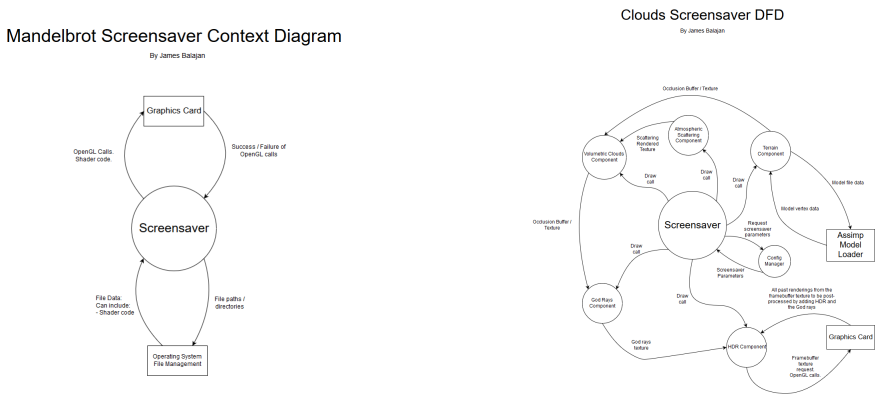


Figure 4.1: *From left to right: Mandelbrot Screensaver Context Diagram, Clouds Screensaver DFD. For full resolution versions refer to appendix*

Furthermore a data dictionary was created to catalogue the variables used in each of the three screensavers.

The data dictionary may be found on page 43. The design diagrams may be found on their respective pages as shown in table 4.1.

Table 4.1: *Page Numbers of Design Documents*

Screensaver	Context Diagram Page No.	DFD Page No.	Structure Chart Page No.	Story board Page No.
Clouds and Mountains	27	28	29	30
Mandelbrot	31	32	33	34
Metaballs	35	36	37	38

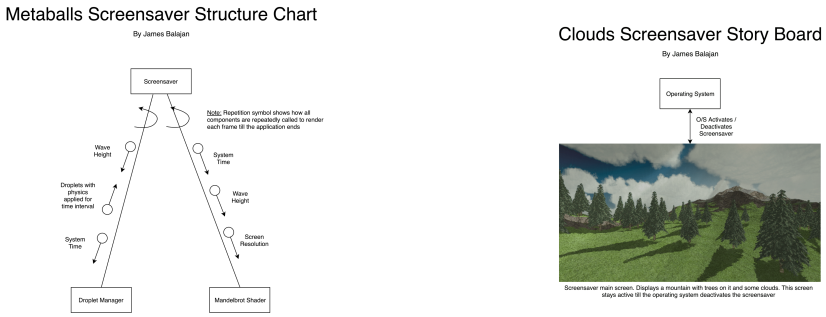


Figure 4.2: *From left to right: Metaballs Screensaver Structure Chart, Clouds Screensaver Storyboard. For full resolution versions refer to appendix*

4.2 Modularity and Design Patterns

4.2.1 Clouds and Mountains Screensaver

When designing the clouds and mountains screensaver, careful attention was put into making an effective modular system, with each module only completing a certain task. By using these object-oriented design principles, each module is effectively able to be removed or added back to the screensaver easily, or even used for different purposes than initially intended so long as it is provided the required inputs. Each module adheres to the *Component* design pattern as defined in (Nystrom, 2014), meaning that they are classes which abstract the lower-level behaviour of higher level classes, to ensure modular design, clean code and to reduce the coupling between different classes, meaning that the modules may be moved about and altered without causing the code to break.

A detail not shown in the data flow diagram is the use of the *Flyweight* design pattern as defined in (Gamma, 1995), which was implemented to optimize RAM usage. The *Flyweight* design pattern allows for the screensaver to only need to store the model data once on RAM, whilst simultaneously allowing for customization of the trees so that the forest does not look repetitive. If each instance of a tree had its own model, the amount of data that would need to be stored on RAM would grow '$O(n)$ with the number of trees drawn to the landscape. To put that in perspective, the tree model used in the screensaver uses 86886 vertices to draw the leaves alone. If a copy of this data were stored for every instance of a tree drawn, the screensaver would quickly run out of RAM. A pictorial representation of a forest without the *Flyweight* design pattern compared with a forest with the pattern is shown is show in figure 4.3. Notice how the optimized version does not contain duplicate data. Each tree instead references the same model data, altering it based on its parameters.

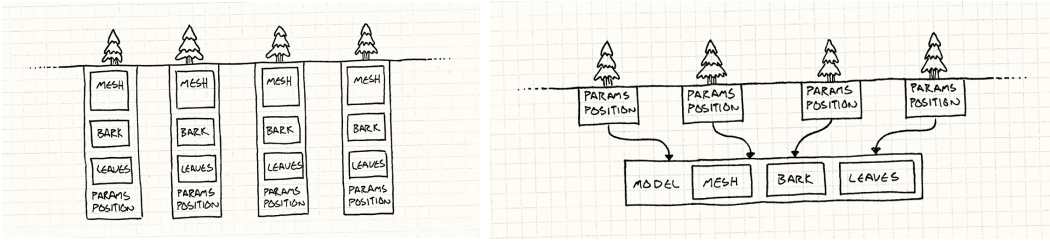


Figure 4.3: *A forest's storage without the Flyweight design pattern vs. with it. Sourced from (Nystrom, 2014)*

This design pattern may be extended to even reducing the number of OpenGL draw calls, and vertex data being sent to and from the graphics card, using a feature in OpenGL called *instancing*, which allows for only one draw call to be used to render all of the trees. Running OpenGL calls such as `GL.DrawElements` is more expensive than the actual drawing of the vertices, as OpenGL needs to always undertake the necessary preparations to make the data ready for consumption by the GPU, and must also transfer the data slowly along the bus from the CPU to the graphics card. (Vries, 2016). By compressing all of these draw calls into a single instance draw call, the data may be transferred to the graphics card in a single, contiguous packet, avoiding all of the overhead associated with multiple OpenGL calls. In this way, the design of the product was not only carefully created to ensure a clear and logical method of viewing the code with limited coupling, yet was also able to optimize the actual product itself.

4.2.2 Mandelbrot Screensaver

The mandelbrot screensaver's calculations occurred entirely on a shader program on the graphics card. However, some efforts were still made to ensure modularity within the program, for even simple programs can benefit from it. For instance, as seen in the data flow diagram (pg. 32), the configuration manager is placed in its own module which reduces the coupling between the actual screensaver itself, and its settings.

Furthermore, a more subtle form of decoupling occurs, in how the graphics code is separated from the management class in that it is stored in a shader. The two of them communicate using an GLSL feature called *uniforms*, which are variables that are passed from the CPU to the graphics card via the bus every frame drawn. For instance, variables which are managed by the broader *Screensaver* module, such as the system time, screen resolution, seed for the first zoom point and selected mandelbrot fractal palette are passed into the shader as defined by the uniforms:

```
uniform vec3 iResolution;
uniform float iTime;
uniform int zoomPointSeed;
uniform int selectedPalette;
```

4.2.3 Metaballs Screensaver

The metaballs screensaver is similarly designed to the mandelbrot screensaver, with the graphics calculations largely confined to its shader program. Like the mandelbrot screensaver, the metaballs screensaver too decouples its configuration manager from the main class. However it also decouples the physics calculations for each of the droplets into a droplet manager class as seen in the data flow diagram (pg. 36).

The metaballs screensaver shader is also decoupled from the main metaballs class, with it passing the center and radii of each droplet to the shader, as well as the height / amplitude

of the waves, the system time and the screen resolution, as defined by the uniforms:

```
uniform vec3 iResolution;
uniform float iTime;

uniform float waveHeight;

...

uniform vec3 dropletCenters[NUM_DROPLETS];
uniform float dropletRadii[NUM_DROPLETS];
```

4.3 Influence of Survey Responses on Design and Direction

When creating the final designs for the screensaver suite, the project utilized user feedback based on iterative prototypes of the product that had been made. For instance, the decision to implement a configuration manager for the screensaver suite was derived from Arjun Malik's second survey response (pg. 70) as well as Julian Peen's second survey response (pg. 76). This configuration manager manifests itself in the final data flow diagram for each of the three screensavers, as a "Config Manager" process.

The surveys also showed that personal preference still has an effect on screensaver choice, even when the screensavers are designed to be universally appealing. Samuel Stacy's second survey response (pg. 73) showed that he did not dislike the mandelbrot fractal screensaver because of any technical or visual issue, yet simply due to personal preference. On asking him in person which screensaver he preferred the most, he said it was the clouds and mountains screensaver. This differed from other survey responses like, for instance, Ms Saki's (pg. 67), who stated that the metaballs screensaver was her favourite. This may have to do with people's subjective perceptions of beauty, for Samuel prefers the more realistic screensaver based on natural beauty, whilst some others prefer the abstract mathematical beauty of the mandelbrot fractal or metaballs. In this way, by creating a suite of three screensavers which contain both real and abstract beauty, the product is able to appeal to both of these preferences.

One suggestion which has been suggested to be added is the option for adding music to the screensaver, as shown in Ms Saki's survey response (pg. 67). This idea is very original, as screensavers have never, in the scope of the market survey conducted, included the option for playing music. However, there do exist some concerns regarding this feature's utility, as screensavers typically run when the user is not at their computer, meaning that there would be no one to receive the music, except for the user when they return from whatever activity removed them from the computer. Furthermore, concerns surrounding time are paramount as the project is beginning to reach its conclusion, as the screensavers must be tested on other architectures and fixed if any issues are present, meaning that there may not be enough time to both implement this new feature and complete the testing. If this feature is not added to the final product, and it is likely that it would not be, a future experiment should be conducted to observe both the market and technical feasibility of a screensaver which employs both visual and auditory stimuli. If this experiment goes well, auditory features may either be incorporated into the current product, or a new screensaver may be created which would better showcase the potential of an audio-visual screensaver.

All of the surveys conducted may be found in the appendix.

4.4 Configuration Windows

One excellent suggestion from Julian Peen's second survey (pg. 76) was to enable the user to change the settings of the screensaver.

The configuration windows were designed with usability and inclusivity in mind. All labels were given large font sizes to assist people with mild visual impairments with seeing the text. The sliders' values are displayed in the label as they change dynamically, allowing for the user to have a quantitative idea of how their settings have changed. The configuration window screen design may be found on page 39.

On quitting the configuration window without saving, the application will open a message box, asking the user if they want to quit without saving their settings to prevent loss of settings. The appropriate question mark icon is used in this message box, as it is asking the user for input, rather than displaying an error message or a warning.

Figure 4.4: *Message Box Which Appears on Not Saving Configuration Settings*

Localization was designed to work by storing the configuration window text in a `.resx` file for each language. When the language is changed, the configuration window would then swap the current resource file for the one corresponding to the new language whilst reloading the form to reload the text on screen.

Chapter 5

Implementation

5.1 Mandelbrot Screensaver

The Mandelbrot fractal, also known as the mandelbrot set, is the set of complex numbers (c) which do not diverge when iterated through the mandelbrot function ($z = 0, 1, 2, 3, ...$). The mandelbrot function is defined in figure 5.1.

$$f_c(z) = z^2 + c$$

Figure 5.1: *Mandelbrot Equation*

The mandelbrot fractal is of particular interest for being a screensaver due to its properties of possessing an infinitely complex structure, which displays a profound mathematical beauty.

5.1.1 Point Mapping

The mandelbrot equation, as seen above, is not of a complex nature, and implementing it is trivial. The first consideration that must be made is how to map the screen pixel coordinates to the complex plane. A complex number consists of both an real component and an imaginary component. One may map the real component of the complex number to the x-axis and the imaginary component of the number to the y-axis ($c = x + iy$). This is illustrated in figure 5.2

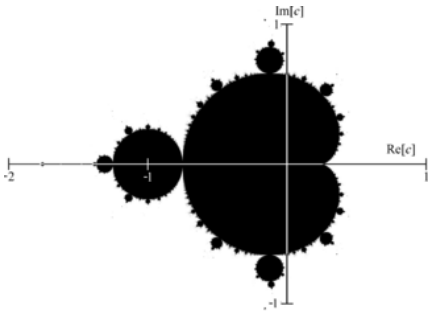


Figure 5.2: *Real and Imaginary Component Mapping to Cartesian Plane*

However, the screensaver must be able to zoom into the fractal, so the screen coordinate mapping to the complex plane must adjust based on the zoom value. This was solved by the following code:

```
vec2 c = currZoomPoint.point + screenPos * zoom;
```

The above mapping code adjusts the mapping of pixel coordinates to complex numbers by multiplying by a zoom value. This is then added to the coordinates of the point the program wishes to zoom into to shift the pixel coordinate mapping so that its center would be the zoomed into point.

5.1.2 Colouring

The mandelbrot fractal could be coloured monochrome, with black indicating that the pixel corresponds to a number which is within the set, and white indicating that the pixel corresponds to a number outside the set, however this would lead to a very dull screensaver. Instead, colours were defined based on inputting the integer iteration count into a palette function. However this resulted in a banding effect which is unappealing to the eye. The solution is to use the smooth iteration count formula (Quilez, 2016).

$$sn = n - \log_2 \log_2(z_n \cdot z_n) + k$$

Figure 5.3: *Smooth Iteration Count Formula from (Quilez, 2016)*

Figure 5.4 contrasts the smooth and discrete mandelbrot fractals. Notice the banding effect at the bottom which is mitigated at the top.

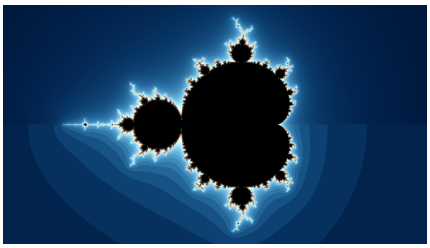


Figure 5.4: *Smooth vs. Discrete Mandelbrot Fractals from (Quilez, 2016)*

5.2 Metaballs Screensaver

The metaballs screensaver attempts to emulate the appearance of droplets falling into the ocean using the organic looking, mathematical object of the metaball, created by Jim Blinn. The screensaver uses a rendering technique called signed distance function raymarching or sphere tracing.

5.2.1 Raymarched Scene

A sphere traced scene is defined by a mathematical signed distance function (SDF). A signed distance function returns the distance from a point in euclidean space ($\vec{p} : (x, y, z)$) and the nearest point on the surface of any object in the scene. For instance, the signed distance function of a sphere is defined as $f(\vec{p}) = ||\vec{p}|| - 1$.

Each droplet was defined as a sphere with a certain radius and position. The wave SDF was defined as a plane SDF distorted by the wave height function from (afl_ext, 2017).

To create the final scene SDF and exhibit the blending metaballs behaviour meant to emulate water, the results of querying point $\vec{p}$ in every object's SDF are then passed through the smooth union operation from (Quilez, 2013). The scene SDF function is shown below:

```
float sceneSDF(in vec3 pos) {
        float blendingFactor = 7.0;
        float dist = wavesSDF(pos);
        for (int i = 0; i < NUM_DROPLETS; ++i) {
                dist = smoothUnion(dist, rainSDF(pos, dropletRadii[i],
                        dropletCenters[i]), blendingFactor);
        }
        return dist;
}
```

5.2.2 The SDF Raymarching Algorithm

Sphere tracing is like raytracing, in how a grid is placed infront of the camera and rays are sent from the camera through each point in the grid, with each grid square corresponding to a pixel in the output image. This is illustrated in figure 5.5.

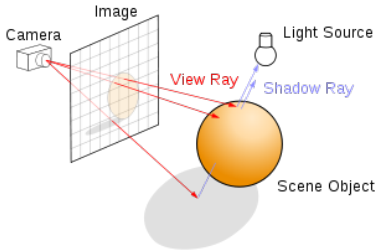


Figure 5.5: *Ray tracing from (Henrik, 2008)*

However, as opposed to conventional raytracing which determines the point of intersection between the cast ray and scene geometry through doing a series of geometric intersection tests, raymarching "marches" along the ray. What this means is that the SDF scene is queried at the beginning of the ray, asking if the point is inside the scene surface, or mathematically phrased as "does the SDF evaluate to a negative number at this point". If not, then the point is moved along the ray by the distance returned from evaluating the SDF. This is repeated till the point does evaluate as being negative, indicating that the ray has collided with an object, or a maximum threshold of increments is exceeded, indicating that no object was collided with. This is known as sphere tracing, and is illustrated in figure 5.6.

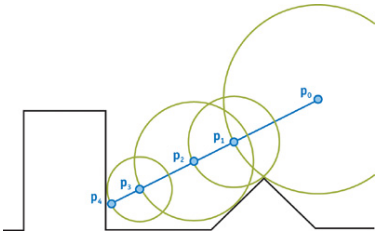


Figure 5.6: *Sphere tracing illustrated from (Pharr, 2005)*

The advantages of using the raymarching technique as opposed to rasteurized or traditionally raytraced graphics are that it allows for the efficient definition of complex scenes through the use of mathematical operations such as the smooth union operation, which would require complex mesh adjustments or geometric primitives in the other two techniques. Furthermore, SDF raymarching allows for the accurate simulation of light based phenomena such as reflection and refraction due to how it colours pixels based on casting rays of light.

The water was coloured based on emulating the light's reflection and transmission through it using the Fresnel equation. Rays which do not intersect with any scene objects are given a colour using atmopsheric scattering, a technique discussed in section 5.3. The code is as follows:

5.3 Clouds and Mountains Screensaver

The graphics techniques used in the clouds and mountains screensaver are the most complex of the three, with there existing many research papers on each of them. As of such, the information on them must be heavily condensed to fit in this portfolio. To find out more comprehensive information on each, please visit their sources.

5.3.1 Atmospheric Scattering

Atmospheric scattering involves numerically estimating the values of a series of light scattering equations by marching along a light ray and using the trapezoidal rule. There exist two types of light scattering models which are used to estimate how the light at the point P is modeled, due to the many different types of particles in the atmosphere. These are known as *Rayleigh scattering* and *Mie scattering*. Rayleigh scattering is caused by small molecules in the air whilst Mie scattering is caused by large particles in the air called aerosols (such as ice and dust). (Pharr, 2005)

Phase function

To describe how much light is scattered in the camera's direction, the phase function is used. The phase function used for atmospheric scattering is an adaptation of the Henyey-Greenstein function also used in volumetric raymarching clouds.

$$F(\theta, g) = \frac{3 \times (1 - g^2)}{2 \times (2 + g^2)} \times \frac{1 + \cos^2 \theta}{(1 + g^2 - 2 \times g \times \cos \theta)^{\frac{3}{2}}}$$

Figure 5.7: *The Phase Function from (Pharr, 2005)*

Scattering

The light which reaches the camera is determined by the in-scattering equation, which uses the out-scattering equation in its calculation. The out-scattering equation is the integral mentioned earlier which must be approximated using the trapezoidal rule.

$$t(P_aP_b, \lambda) = 4\pi \times K(\lambda) \times \int_{P_a}^{P_b} exp(\frac{-h}{H_0})ds$$

Figure 5.8: *The Out-Scattering Equation from (Pharr, 2005)*

The in-scattering equation describes how much light is added to a ray through the atmosphere due to light scattering from the sun.

$$I_v(\lambda) = I_s(\lambda) \times K(\lambda) \times F(\theta, g) \times \int_{P_a}^{P_b} (exp(\frac{-h}{H_0}) \times exp(-t(PP_c, \lambda) - t(PP_a, \lambda)))ds$$

Figure 5.9: *The In-Scattering Equation from (Pharr, 2005)*

By evaluating the in-scattering equation for each pixel on the screen, the colour due to atmospheric scattering can be approximated. This evaluation was implemented in a graphics shader programmed in GLSL.

5.3.2 Volumetric Raymarching

Cloud Modelling

The volumetric cloud system is modelled using volumetric raymarching in real time through a 3D texture of fractal Brownian motion (FBM) noise. FBM is the sum of a series of octaves of noise, each with higher frequency and lower amplitude. Perlin noise is commonly used for this purpose, however it typically fails to describe, the round, billowy shapes of cumulus clouds. This issue may be remedied by combining standard Perlin FBM with another flavour of noise called Worley noise, which is normally used for rendering water effects, to give the noise the cauliflower-like look typical of clouds. (Engel, 2016)

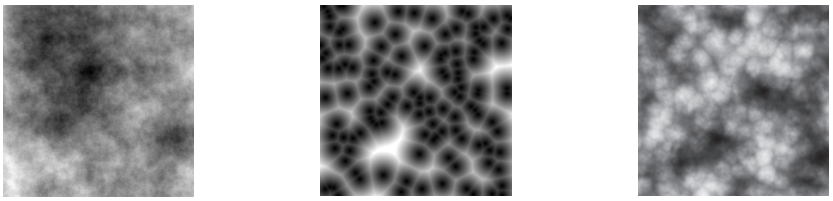


Figure 5.10: *From left to right example images of: Perlin FBM Noise, Worley Noise, Perlin-Worley Noise*

Furthermore, a density height function must be specified to bias, or scale the cloud density value based on height. Each type of cloud possesses its own density height function to describe the different altitudes at which these clouds form at. (Engel, 2016)

Figure 5.11: *From left to right example images of height gradients for: Stratus, Cumulus and Cumulonimbus clouds from (Engel, 2016)*

Finally, a 2D weather texture is generated to specify the positions of clouds in the sky. It is created using low octave noise on the GPU through a compute shader. As GPUs do not possess random number generating capabilities by default, a random number generation function needed to be added which also supported seeding. The function which was used in known as gold noise, and uses the irrationalities of π, $\sqrt{2}$ and ϕ to generate pseudo-random numbers (dcerisano, 2015).

The Algorithm

Now that a model to describe clouds is created, it must be translated to colours on the screen. This is done through the volumetric raymarching algorithm which consists of 4 basic steps:

1. **Ray casting**. Much the same as for SDF raymarching, a ray is cast from the camera through a grid infront of it.
2. **Sampling**. Along the path of the ray in the volume, samples are taken. In the case of the cloud model it samples the transmittance of the clouds.
3. **Shading**. Each sample point is shaded based on its value according to a shading algorithm.
4. **Compositing**. After all the sample points have been shaded, they are composited along the ray of sight to create the final pixel colour.

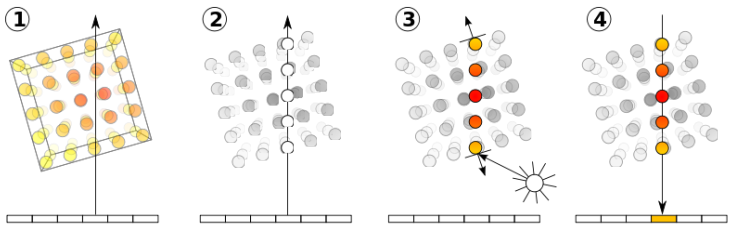


Figure 5.12: *Image illustrating the 4 steps of volumetric raymarching from (Hofmann, 2011)*

Shading

When light enters a cloud, the majority of the light rays spend their time refracting through the water droplets inside of the cloud before scattering out to the eyes. (Engel, 2016) A photon inside a cloud may do one of the three things:

1. Be absorbed by a particle in the cloud. This is known as *extinction* or *absorption*.
2. Exit towards the eye. This is *in-scattering* and was visited when atmospheric scattering was implemented.
3. Exit away from the eye. This is *out-scattering* and was also visited in atmospheric scattering.

The Beer-Lambert law is used to model the probability of these three outcomes.

$$E(b, d_s) = e^{-b \times d_s}$$

Figure 5.13: *Beer-Lambert Law from (Haggstrom, 2018)*

However, Beer's law is an extinction model, meaning that it is concerned with how light energy attenuates over depth. This is ineffective at modelling an important lighting effect

related to in-scattering on the sun-facing sides of clouds, which present themselves as dark edges on clouds when a view ray approaches the direction of the light ray. This is a similar effect to what is seen in powedered sugar, and is hence named the powdered sugar effect. (Engel, 2016)

To account for the powdered sugar effect the amended Beer's-Powder function is created.

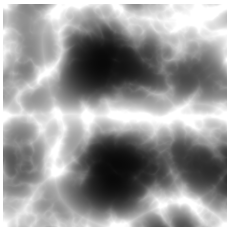
$$E = 2e^{-d}(1 - e^{-2d})$$

Figure 5.14: *Beer's-Powder function from (Engel, 2016)*

Using this (along with the Henyey-Greenstein function which is not covered for the sake of brevity), the cloud samples may be shaded.

5.3.3 Procedural Terrain

The terrain mesh is generated from a procedurally generated height map. The height map is generated using *RidgedMultiFractal* noise from the *SharpNoise* library. Samples are taken along the height map periodically, to determine the vertices of the terrain mesh. These points are then joined together into triangles which map be rendered to the screen through OpenGL. Tree models are randomly placed along the terrain. The terrain textures implement normal/bump mapping to give them depth when shaded using Blinn-Phong shading.



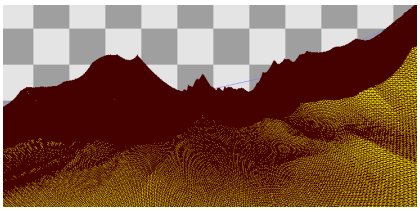


Figure 5.15: *From left to right: Typical heightmap, Mesh viewed in RenderDoc Debugger*

5.3.4 God Rays

God rays, also known as crepuscular rays, are the rays of light which are seen as it scatters off of particles in the air and into our eyes. They derive their heavenly name due to how it seems, when the clouds part to reveal the sun, that God is shining down. They are implemented as a post-process effect on an occlusion texture. This occlusion texture specifies where on the screen the sun is occluded from the camera. The algorithm solves the following equations for each texel of the occlusion texture.

$$L(s,\theta) = L_0 e^{-\beta_{\text{ex}} S} + \frac{1}{\beta_{\text{ex}}} E_{\text{sun}} \beta_{\text{sc}}(\theta)(1 - e^{-\beta_{\text{ex}} S})$$

$$L(s,\theta,\phi) = \sum_{i=0}^{n} \frac{L(s_i,\theta_i)}{n}$$

Figure 5.16: *Equations representing the recurrent determination of the amount of light absorbed from the point of emission to the view point as well as the amount due to light scattering into the path of the view ray from (Nguyen, 2007)*

Chapter 6

Testing

Testing was completed through close collaboration with users. The user Matthew Richards completed a pilot test of the product, and raised several issues such as the clouds and mountains screensaver not rendering after a few runs, as well as a notable jittering effect with the God rays. Communication was undertaken using the free online platform *Discord*^{TM}, and screenshots of the chat are shown. *Balajanovski* is the developer's username whilst *Lasagenator* is the user's.

Figure 6.1 shows the testing where it was observed that, after the program not working using 1:1 screen resolution ratio, it failed to work using any.

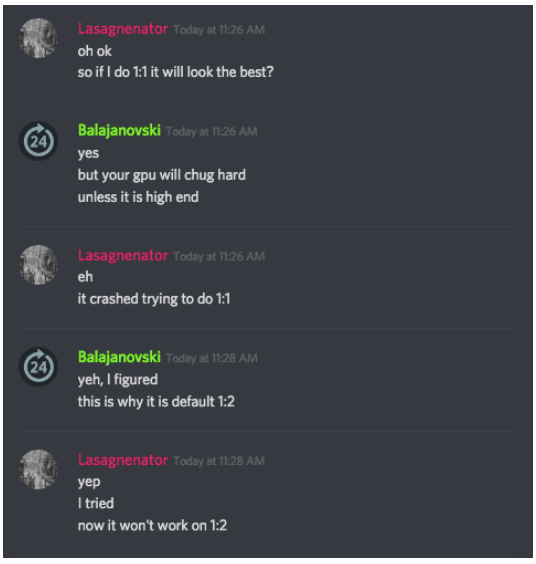


Figure 6.1: *Screensaver not rendering*

All of the screensaver settings are saved in the Windows registry, so the issue may have been caused by a faulty *resolutionRatio* registry value. Matthew was asked to provide a screenshot of his Windows registry, shown in figure 6.2, revealing that the resolution ratio is of the expected value, meaning the issue must be caused by something else.

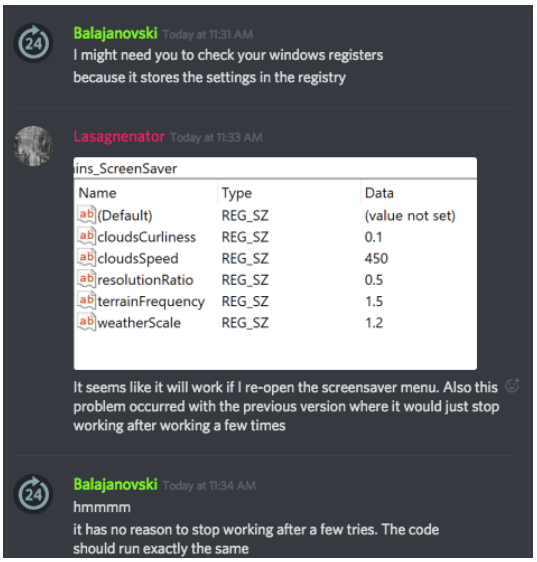


Figure 6.2: *User Windows registry showing expected values*

Compute shaders run on the startup of the screensaver, and it was theorised that the user may have believed that the screensaver was not rendering as the compute shaders were still generating noise. To test this theory, Matthew was asked to allow the screensaver 5 seconds of time to allow it to render. After a few seconds of waiting the screensaver did render, revealing that there was no issue with the screensaver not rendering after successive activations.

However, a new problem was revealed which indicated a graphical bug that lead to the jittering of the God rays. Mention of the problem is shown in figure 6.3.

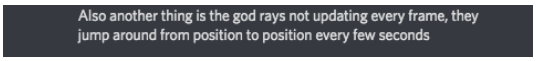


Figure 6.3: *God rays jumping*

This was fixed by increasing the distance of the sun from the camera position by upscaling the sun position unit vector. As the sun was too close to the camera, when it was projected to the screen through matrix multiplication with the camera's view and projection matrices, small changes in the position would not be detected as the mantissa of the floats would be truncated. Only once the change in the position vector became large enough would the change be observable, leading to the God rays seemingly jumping about. Once this change was made, the bug ceased to be exist, as shown in figure 6.4.

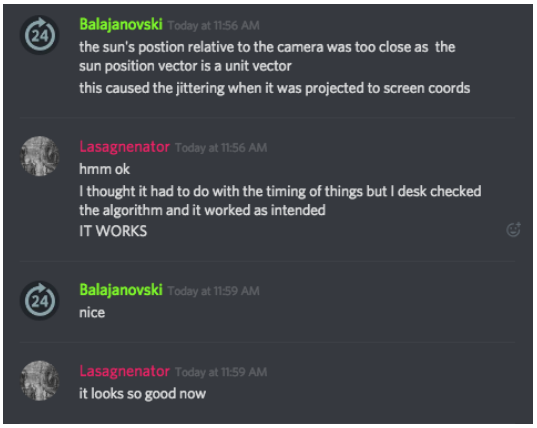


Figure 6.4: *God ray jittering bug fixed*

Chapter 7

Evaluation

To decide if the project is a success, it must first be compared to the problem definition requirements to ensure that it has satisfied the project's demands. To summarise, these requirements are to:

1. Create a suite of screensavers which effectively utilise advanced graphics techniques to generate visual appeal.

2. Run effectively (20-30 FPS) on an average computer's hardware.

3. Allow for users to customise their screensavers through a settings GUI.

4. Complete the project with minimal cost.

5. Complete the project on time.

6. Attract a global community of users through including a Russian locale as well as an English locale.

Based on the positive feedback from the surveys conducted, which may be found in the appendix, it is reasonable to suggest that the screensavers possess visual appeal, satisfying requirement one.

Requirement two however is not so simple. The second and third screensavers run effectively between 20 and 30 frames per second (FPS) on maximum resolution, however the first is only able to, on the developer's and pilot user's hardwares, when the resolution option is decreased. This is likely due to the expensive volumetric raymarching algorithm which must be computed each frame to render the clouds. However, it is reasonable to suggest that the project has satisfied requirement two, as the resolution option is specifically programmed to allow for a greater scope of screensaver usage by allowing for people who do not possess powerful computers to use the screensavers, at the expense of overall quality. The user may simply turn down the resolution to use the screensaver, sacrificing some of the high resolution but still producing a visually appealing result.

Requirement three was satisfied by creating a settings GUI using WPF (Windows Presentation Framework), which would be activated on the user pressing the settings button in the Windows OS screensaver selection screen. The GUI satisfied the inclusivity requirements laid out, using large sans-serif font to aid users with visual impairments in navigating it.

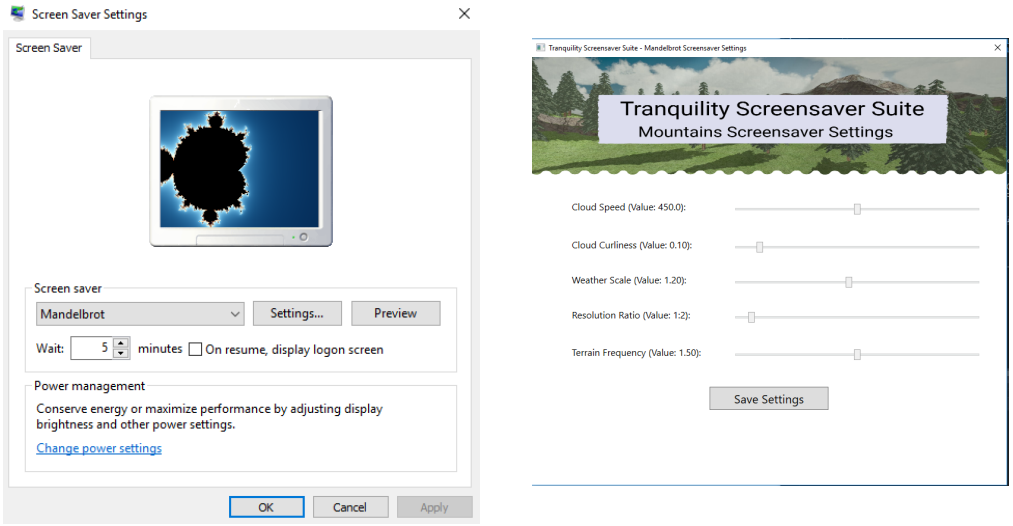


Figure 7.1: *From left to right: Mandelbrot screensaver previewed in Windows screensaver selection GUI, Mountains screensaver Settings GUI*

Since the project was created with no costs other than those incurred by charging the laptop, as the developer voluntarily created the project, criterion four was satisfied.

Furthermore, the implementation and testing stages were completed on the 10th of June, well before the deadline, satisfying criterion five.

Finally, the option to change the locale of the screensaver settings GUI was implemented. The language is able to be changed through a dropdown box in the top-left of the window. This satisfies requirement six.

As all of the project requirements have been satisfied, it is reasonable to suggest that the project was a success.

Bibliography

afl_ext (2017). *Very fast procedural ocean*. [Online; viewed 2nd of June 2019]. URL: https://www.shadertoy.com/view/MdXyzX.

dcerisano (2015). *Gold Noise random static*. [Online; viewed 4th of June 2019]. URL: https://www.shadertoy.com/view/ltB3zD.

Engel, Wolfgang (2016). *GPU Pro 7: Advanced Rendering Techniques*. A K Peters/CRC Press. ISBN: 149874253X. URL: https://www.xarg.org/ref/a/149874253X/.

Gamma, Erich (1995). *Design patterns : elements of reusable object-oriented software*. Reading, Mass: Addison-Wesley. ISBN: 978-0201633610.

Haggstrom, Fredrik (2018). "Real-time rendering of volumetric clouds". MA thesis. Umea Universitet.

Henrik (2008). *Ray tracing (graphics)*. [Online; viewed 2nd of June 2019]. URL: https://commons.wikimedia.org/wiki/File:Ray_trace_diagram.svg.

Hofmann, Florian (2011). *Volume ray casting*. [Online; viewed 6th of June 2019]. URL: https://commons.wikimedia.org/wiki/File:Volume_ray_casting.svg.

Nguyen, Hubert (2007). *GPU Gems 3*. Addison-Wesley Professional. ISBN: 0321515269.

Nystrom, Robert (2014). *Game programming patterns*. United States: Genever Benning. ISBN: 978-0990582908.

Pharr, Matt (2005). *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation*. Addison-Wesley Professional. ISBN: 0321335597.

Quilez, Inigo (2013). *Distance Functions*. [Online; viewed 2nd of June 2019]. URL: https://www.iquilezles.org/www/articles/distfunctions/distfunctions.htm.

– (2016). *Smooth Iteration Count for Generalized Mandelbrot Sets*. [Online; viewed 2nd of June 2019]. URL: http://www.iquilezles.org/www/articles/mset_smooth/mset_smooth.htm.

Vries, Joey de (2016). *Learn OpenGL*. [Online; viewed 7th of May 2019]. URL: https://learnopengl.com/Advanced-OpenGL/Instancing.

Glossary

Assimp Open Asset Import Library is a cross-platform 3D model import library which aims to provide a common application programming interface for different 3D asset file formats.. 25, 26

Atmospheric Scattering Atmospheric scattering, in the context of computer graphics, refers to the modeling and emulation of the process by which light particles are scattered by particles in the atmosphere. Scattering from these particles leads to the blue color of our sky at noon, and its orange color at dawn and dusk. 1

Component (Design Pattern) Allow a single entity to span multiple domains without coupling the domains to each other. (Nystrom, 2014). 9

Coupling In software engineering, coupling is the degree of interdependence between software modules; a measure of how closely connected two routines or modules are. Low coupling is often a sign of a well-structured computer system and a good design, and when combined with high cohesion, supports the general goals of high readability and maintainability.. 9, 10

Flyweight (Design Pattern) A flyweight is an object that minimizes memory usage by sharing as much data as possible with other similar objects; it is a way to use objects in large numbers when a simple repeated representation would use an unacceptable amount of memory. (Gamma, 1995). 10

GLSL GLSL (GLslang) is a short term for the official OpenGL Shading Language. GLSL is a C/C++ similar high level programming language for several parts of the graphic card. With GLSL you can code (right up to) short programs, called shaders, which are executed on the GPU. 10, 17

Mandelbrot Fractal The mandelbrot fractal or mandelbrot set is the set of complex numbers for which the mandelbrot function does not diverge when iterated from, i.e., for which the sequence, etc., remains bounded in absolute value. Its definition and name are due to Adrien Douady, in tribute to the mathematician Benoit Mandelbrot.. 13

OpenGL a cross-language, cross-platform API standard for rendering 2D and 3D vector graphics. The API is used to interact with a graphics processing unit, to achieve hardware-accelerated rendering. 3, 10, 19

Participating Media Participating media are materials which may absorb, emit and/or scatter light. Participating media usually consist of many particles suspended in the air. 23

Physically Based Rendering In its broadest sense, physically based rendering refers to the usage of physics equations to accurately emulate the trajectories of photons when in contact with materials. An example of a physically based lighting model is the

Cook-Torrance model, as opposed to the old, non-physically based Blinn-Phong model. 1

Shader a type of computer program originally run on graphics card to do shading, aka lighting simulation. However, shaders now perform a variety of tasks, and the term now refers to a category of computer programs that are designed to be interpreted and run by the graphics card. 3, 10, 11

Volumetric Clouds Volumetric clouds are rendered using the technique of volumetric raymarching over real-time generated volumes that are designed to emulate the shapes and behaviours of clouds. 1

Volumetric Raymarching Volumetric raymarching is a graphics technique used to accurately render participating media, such as smoke, fog and clouds. 1, 23

Chapter 8

Appendix

8.1 Designs

8.1.1 Clouds Screensaver Context Diagram

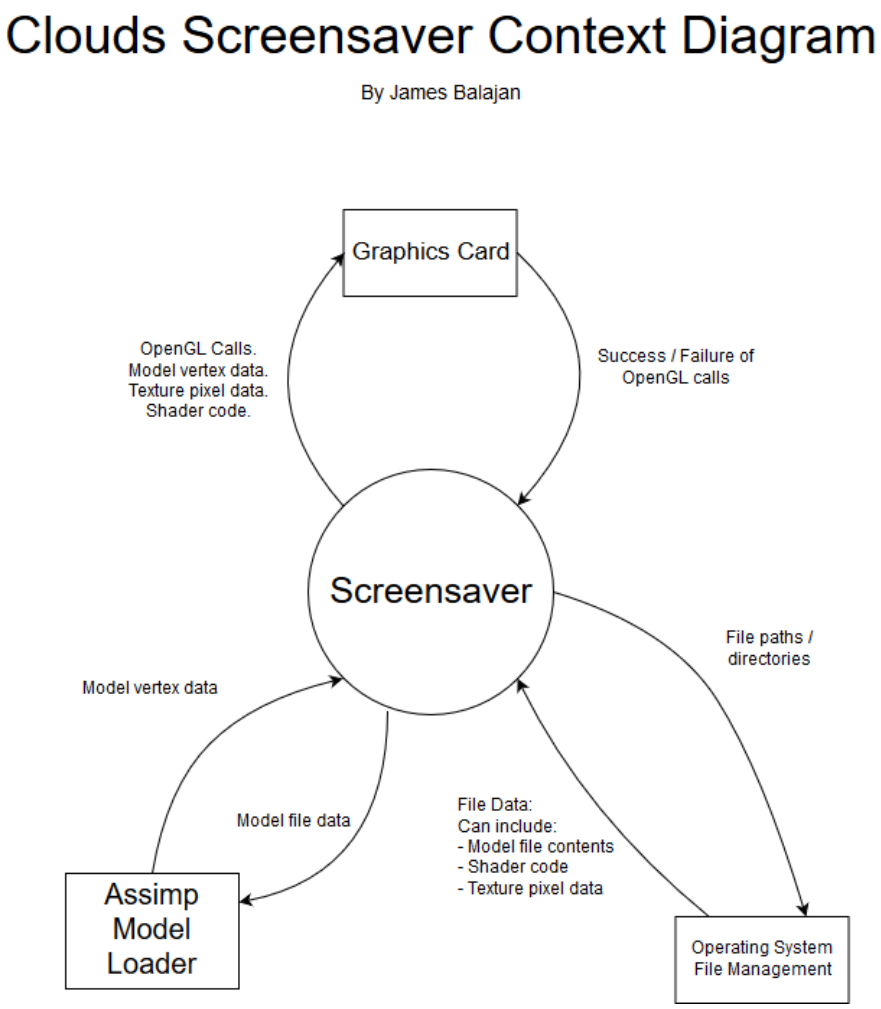


Figure 8.1: *Clouds Screensaver Context Diagram*

8.1.2 Clouds Screensaver Data Flow Diagram

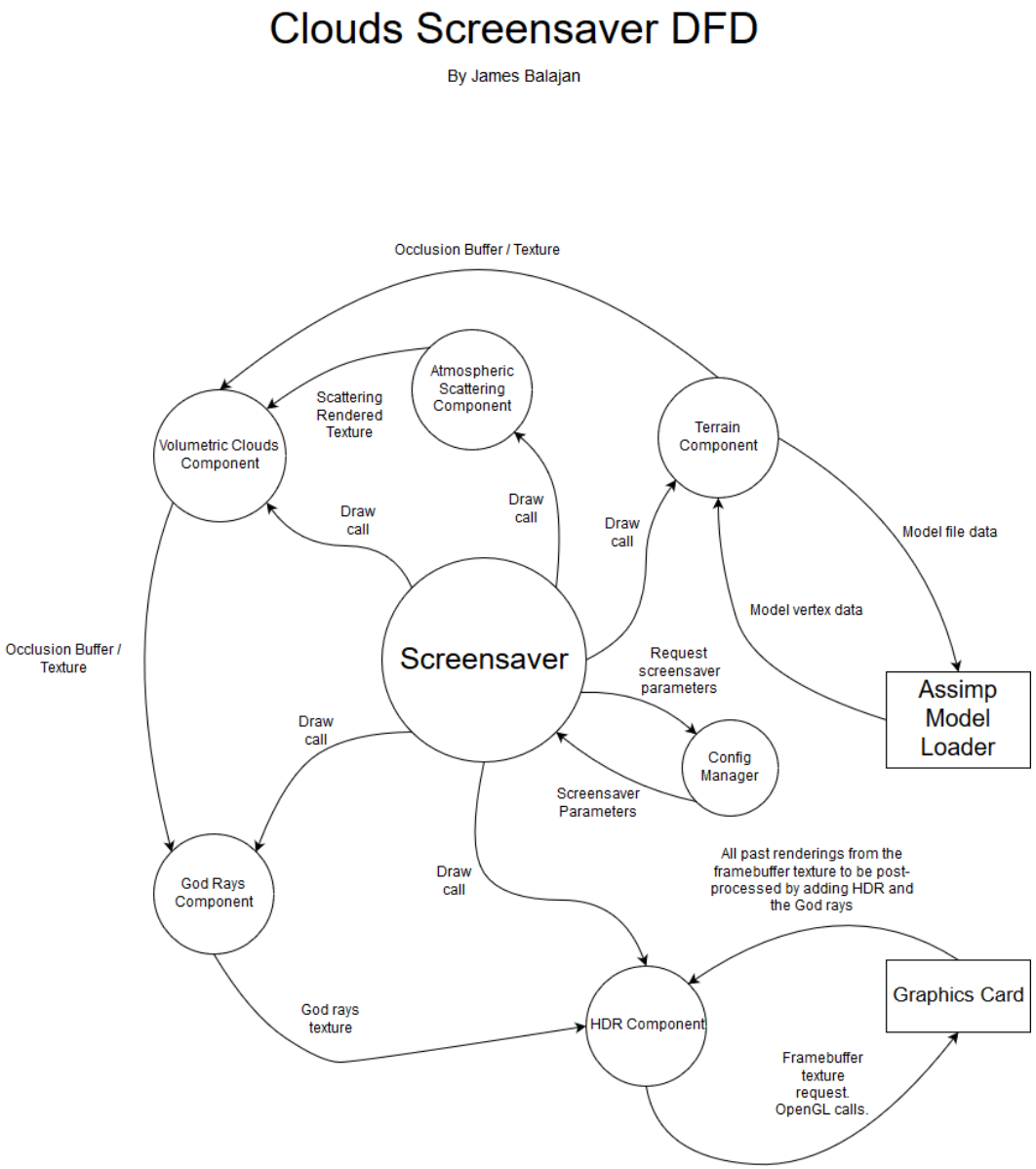


Figure 8.2: *Clouds Screensaver Data Flow Diagram*

8.1.3 Clouds Screensaver Structure Chart

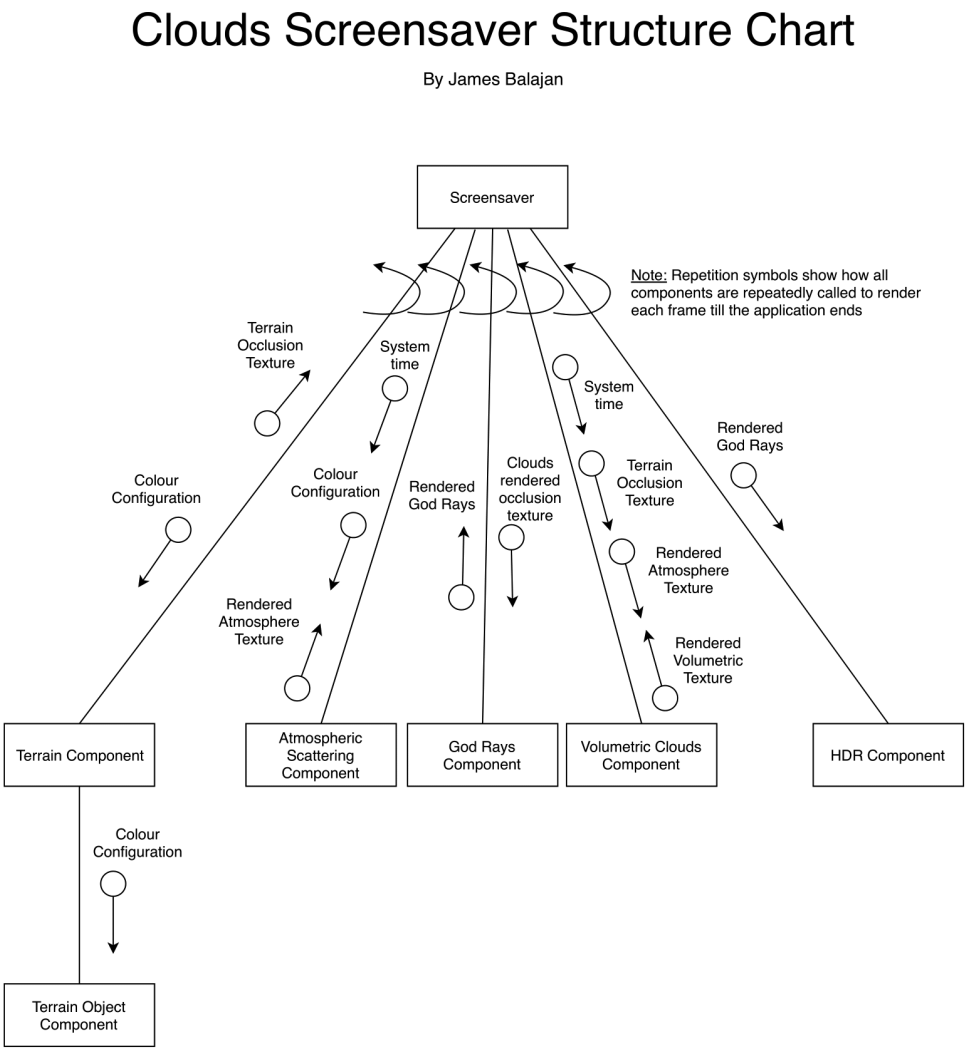


Figure 8.3: *Clouds Screensaver Structure Chart*

8.1.4 Clouds Screensaver Storyboard

Figure 8.4: *Clouds Screensaver Storyboard*

8.1.5 Mandelbrot Screensaver Context Diagram

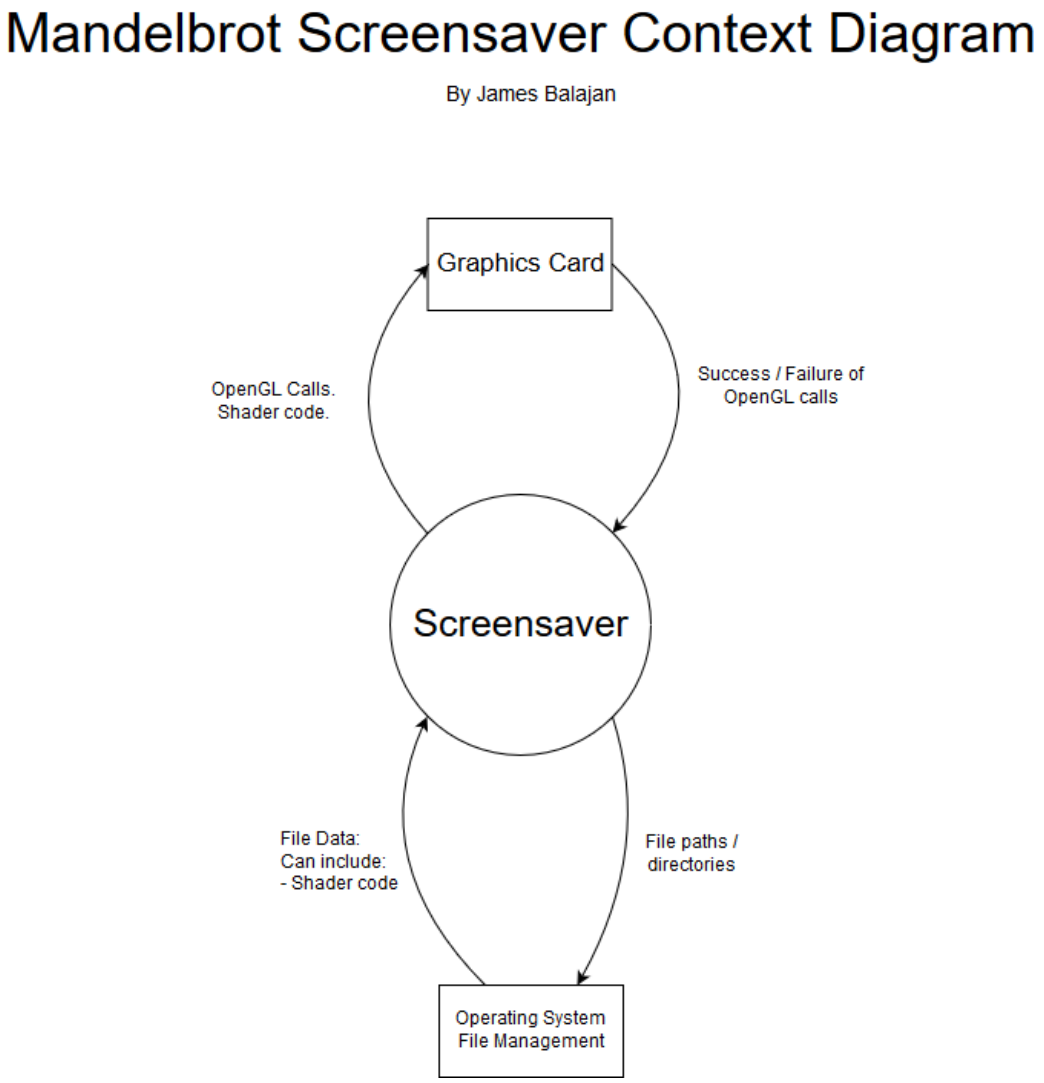


Figure 8.5: *Mandelbrot Screensaver Context Diagram*

8.1.6 Mandelbrot Screensaver Data Flow Diagram

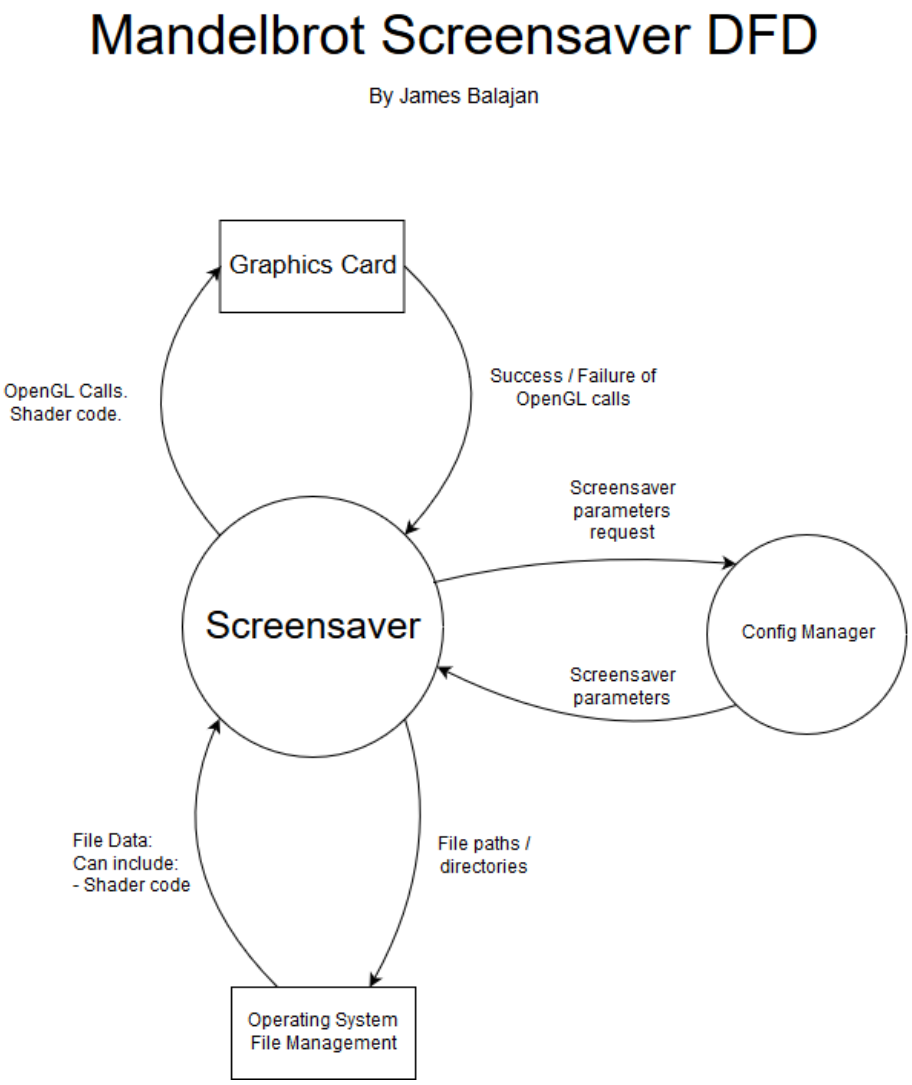


Figure 8.6: *Mandelbrot Screensaver Data Flow Diagram*

8.1.7 Mandelbrot Screensaver Structure Chart

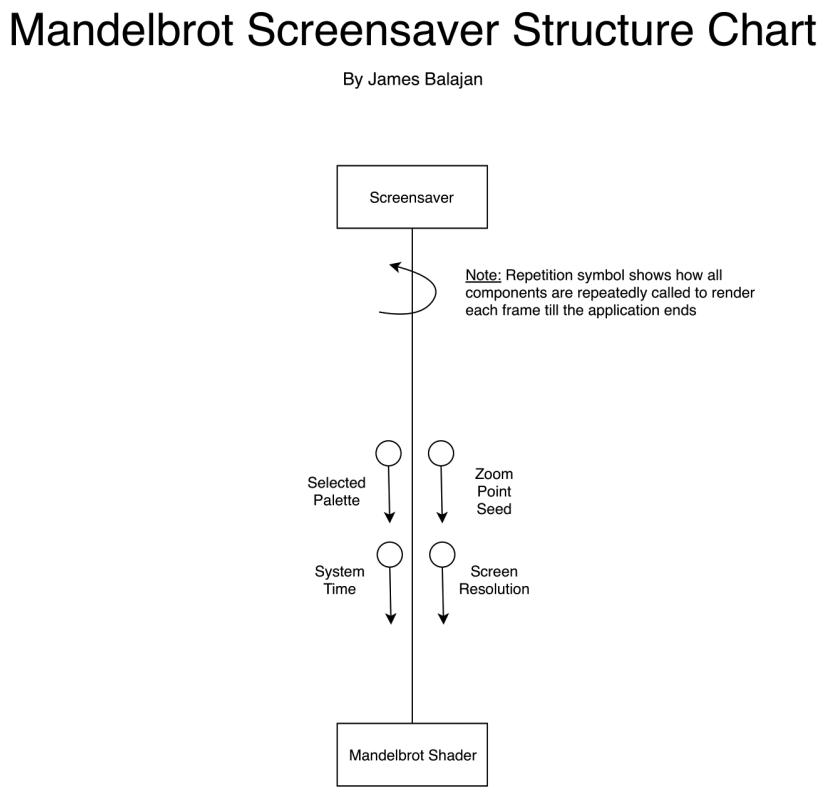


Figure 8.7: *Mandelbrot Screensaver Structure Chart*

8.1.8 Mandelbrot Screensaver Storyboard

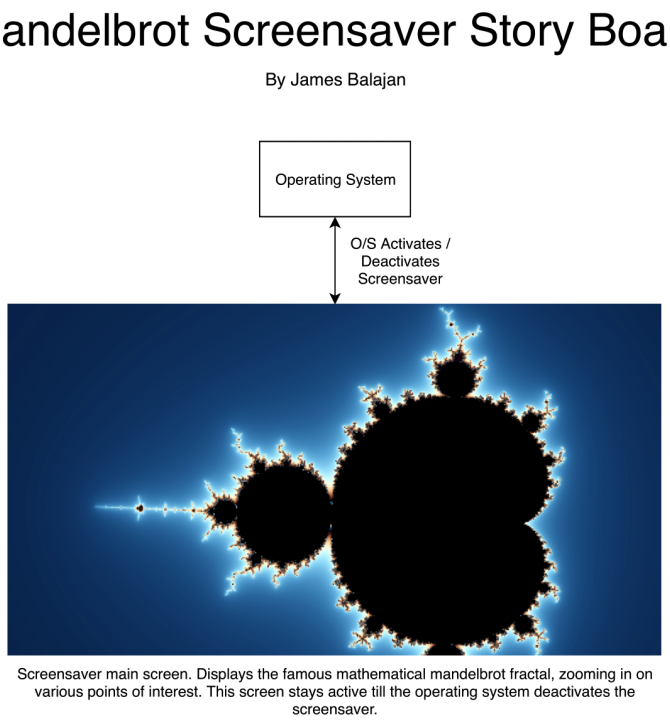


Figure 8.8: *Mandelbrot Screensaver Storyboard*

8.1.9 Metaballs Screensaver Context Diagram

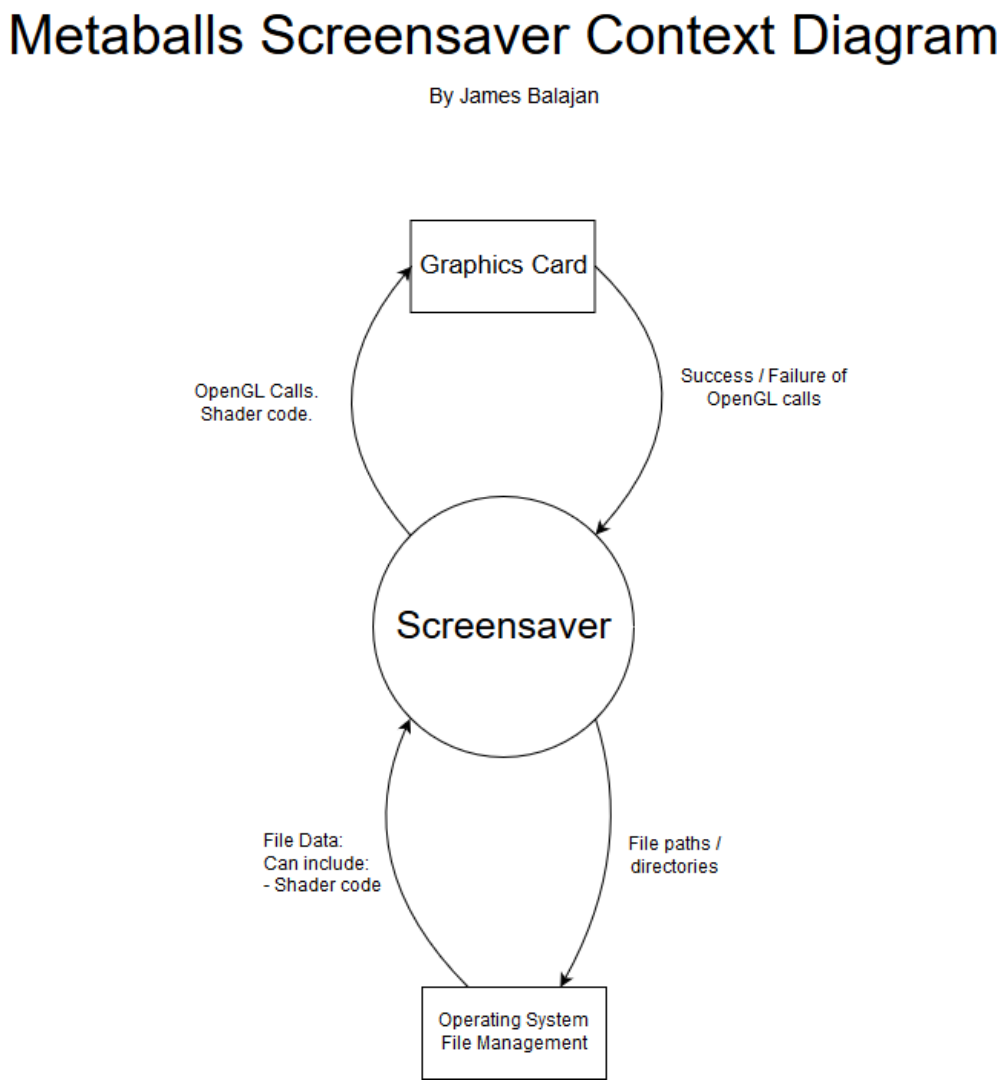


Figure 8.9: *Metaballs Screensaver Context Diagram*

8.1.10 Metaballs Screensaver Data Flow Diagram

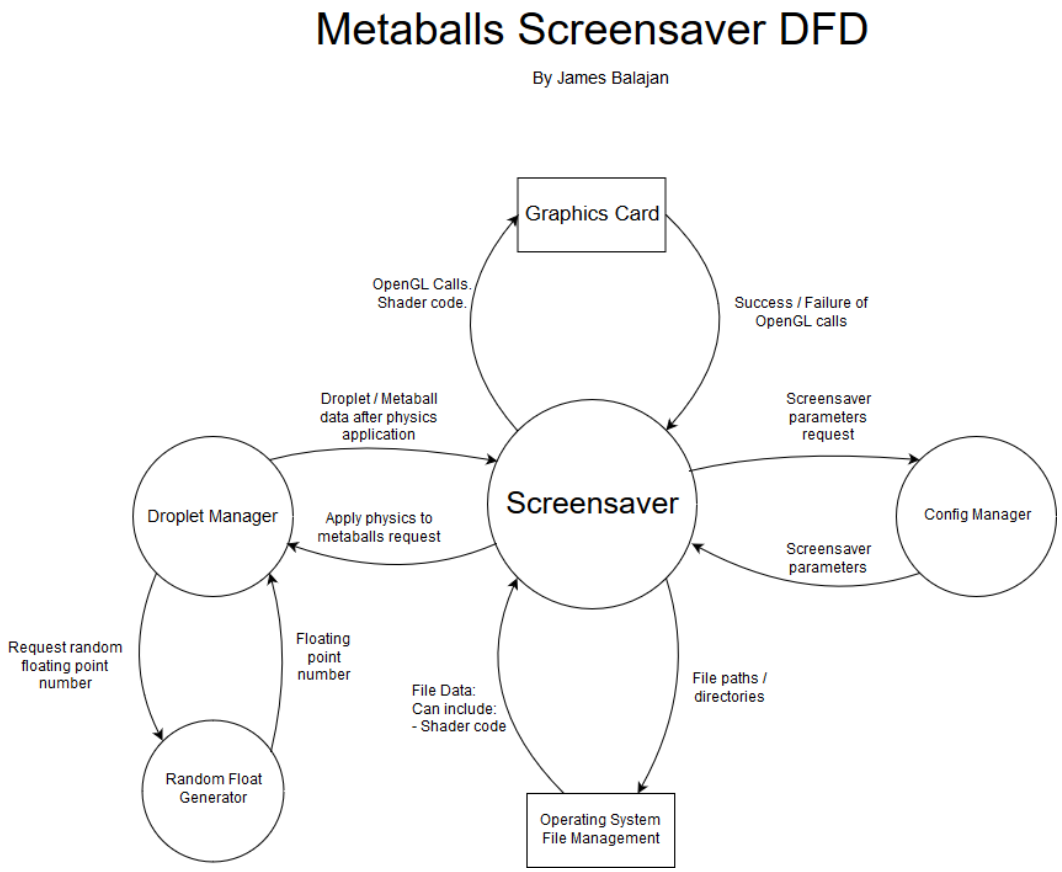


Figure 8.10: *Metaballs Screensaver Data Flow Diagram*

8.1.11 Metaballs Screensaver Structure Chart

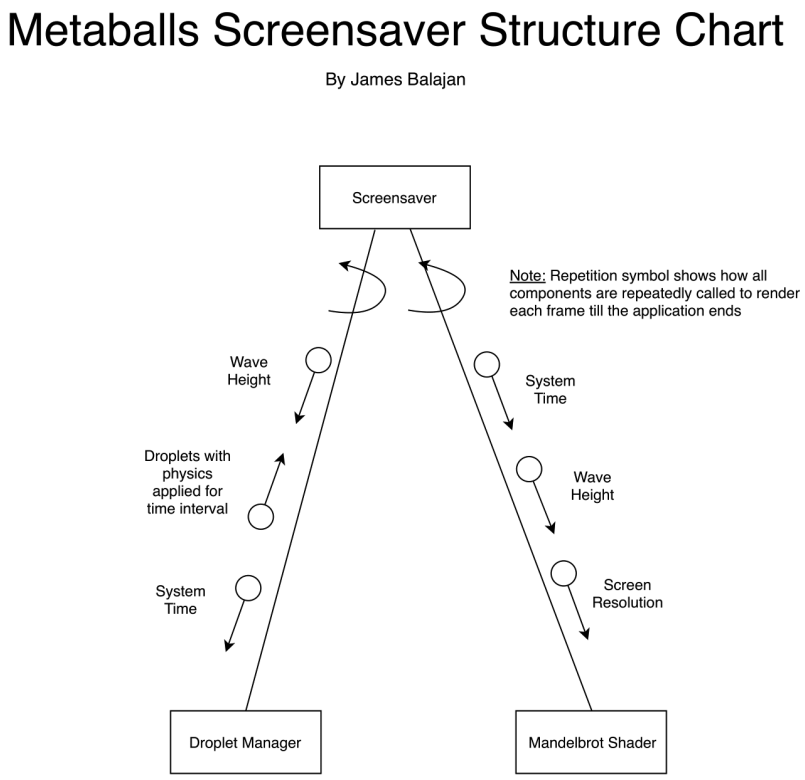


Figure 8.11: *Metaballs Screensaver Structure Chart*

8.1.12 Metaballs Screensaver Storyboard

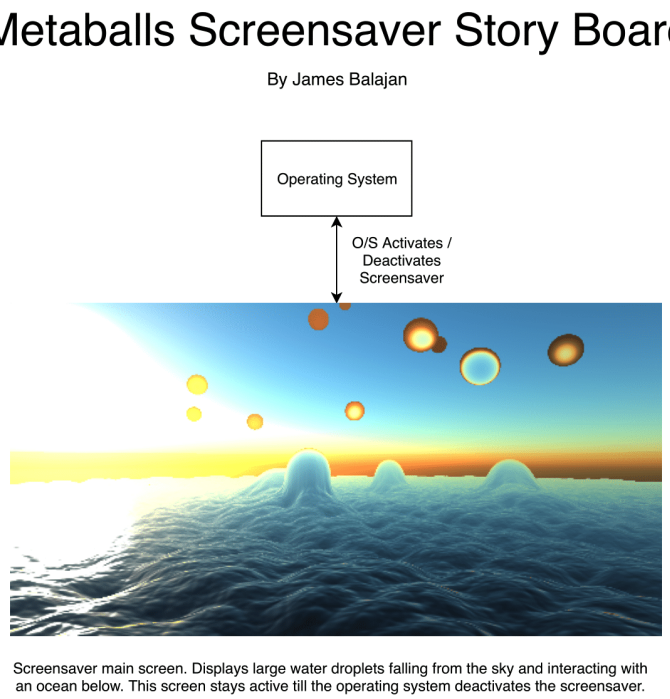


Figure 8.12: *Metaballs Screensaver Storyboard*

8.1.13 Configuration Window Screen Design

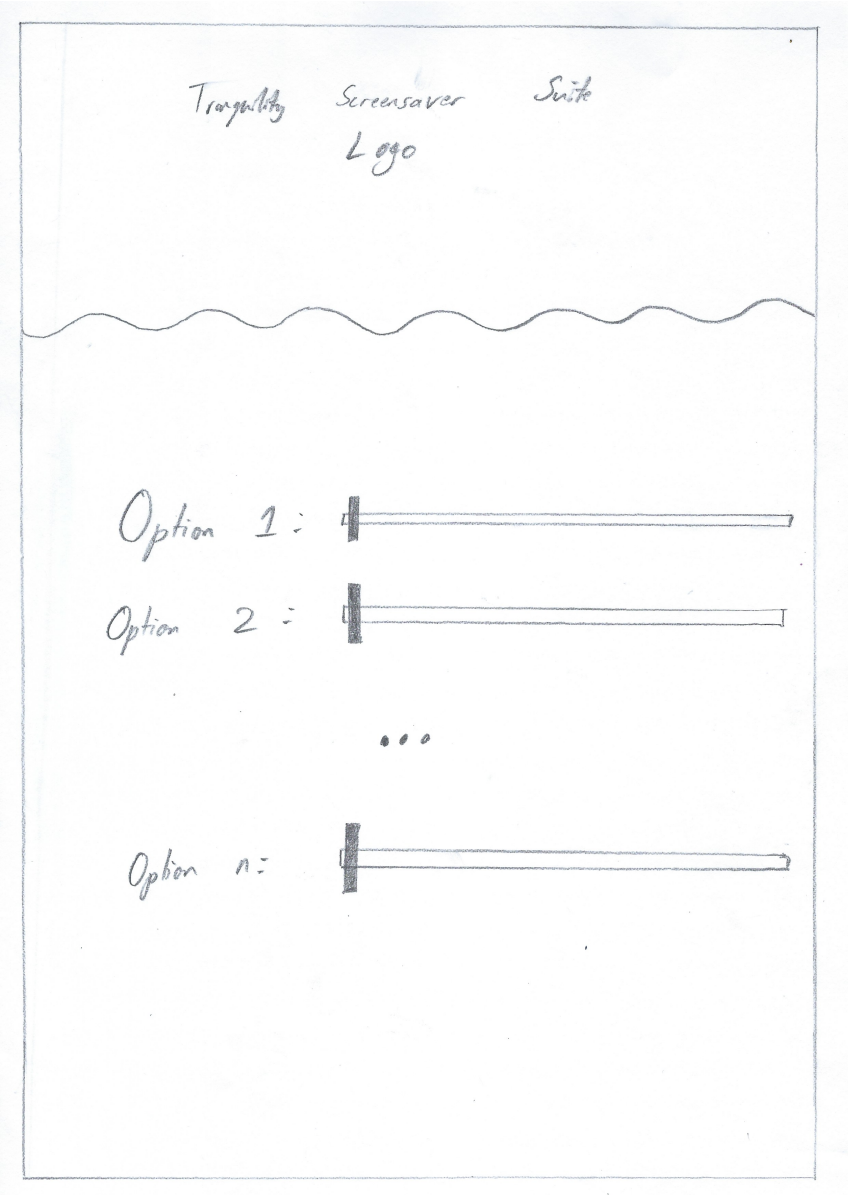
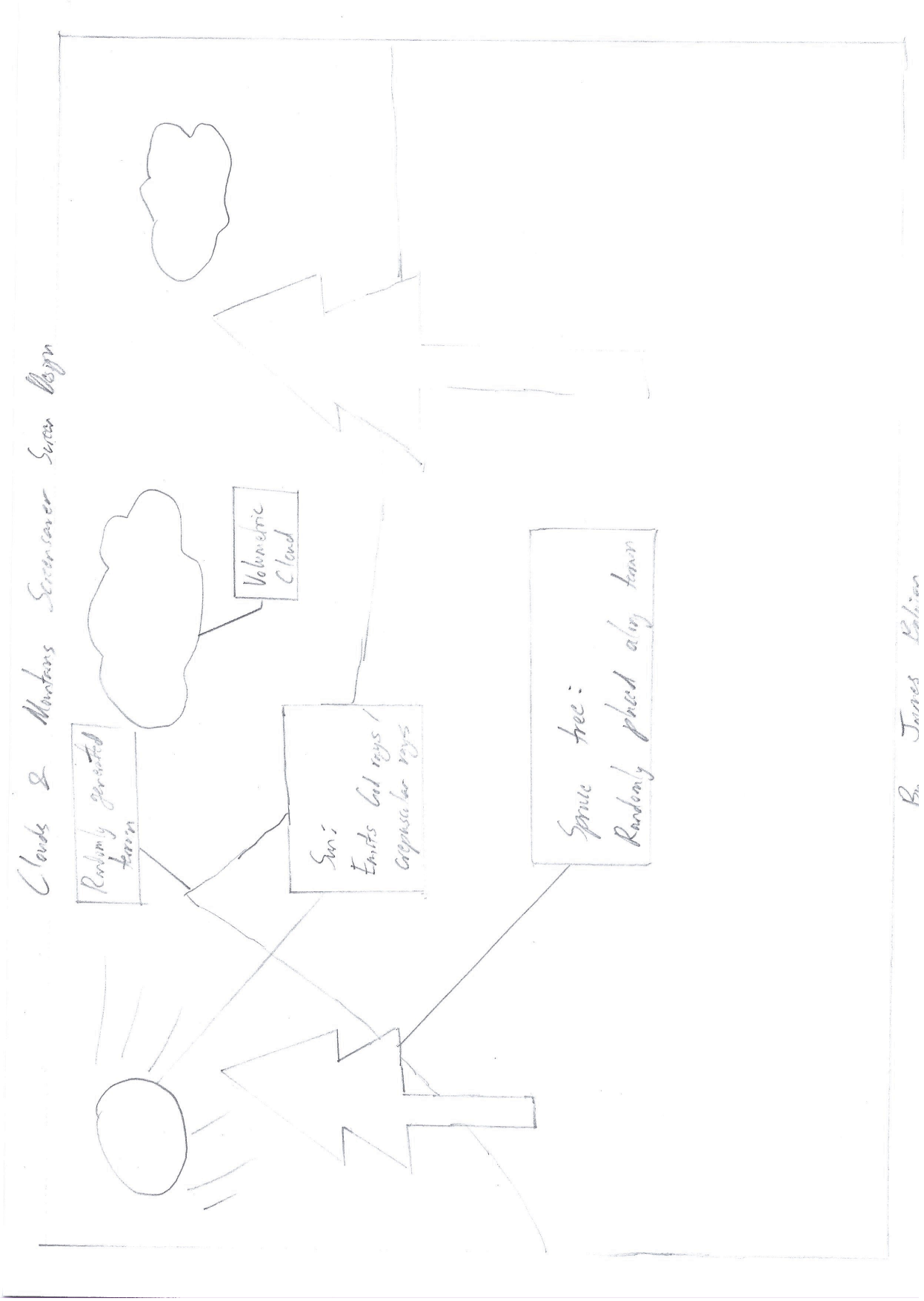
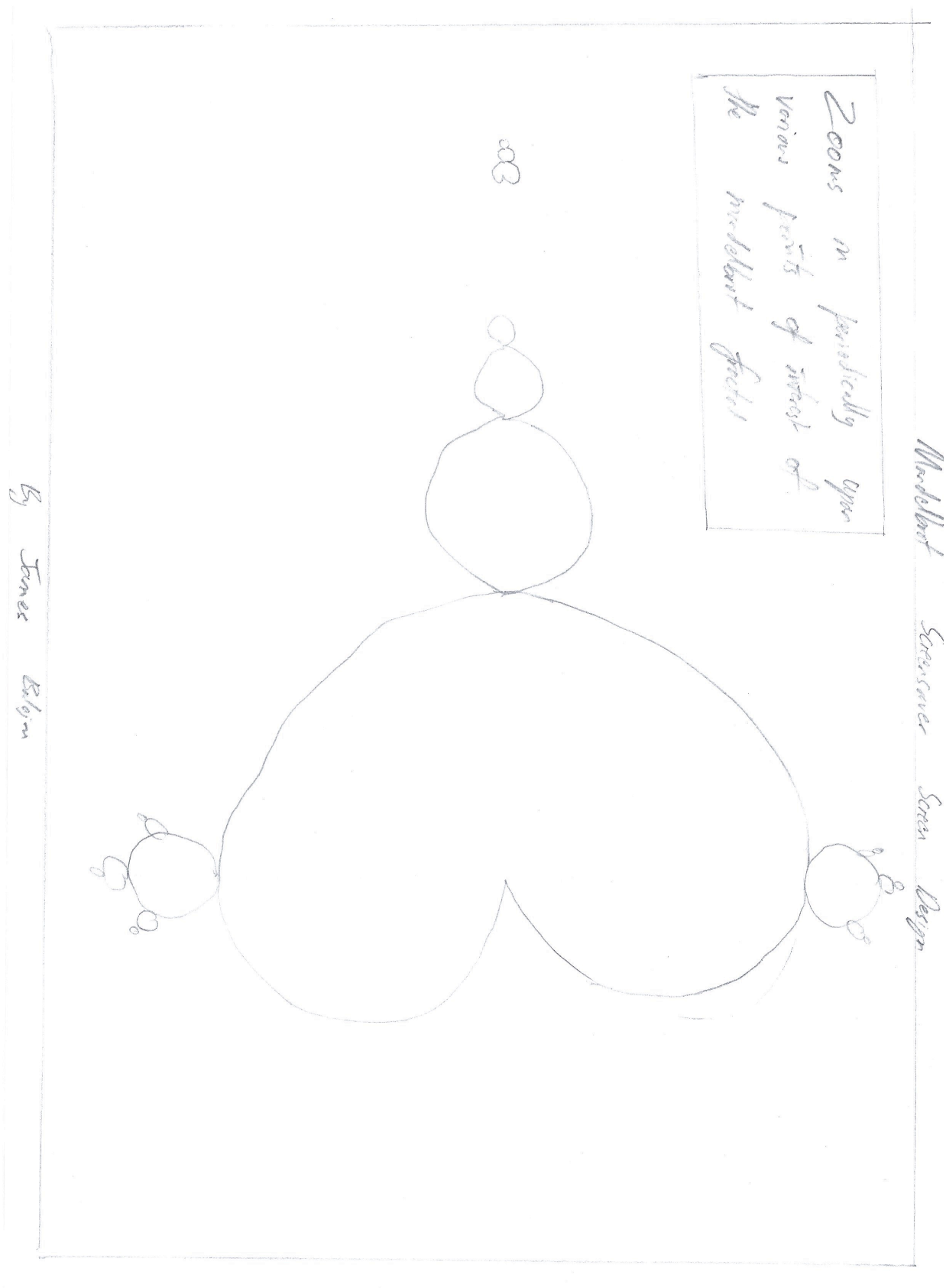


Figure 8.13: *Configuration Window Screen Design*

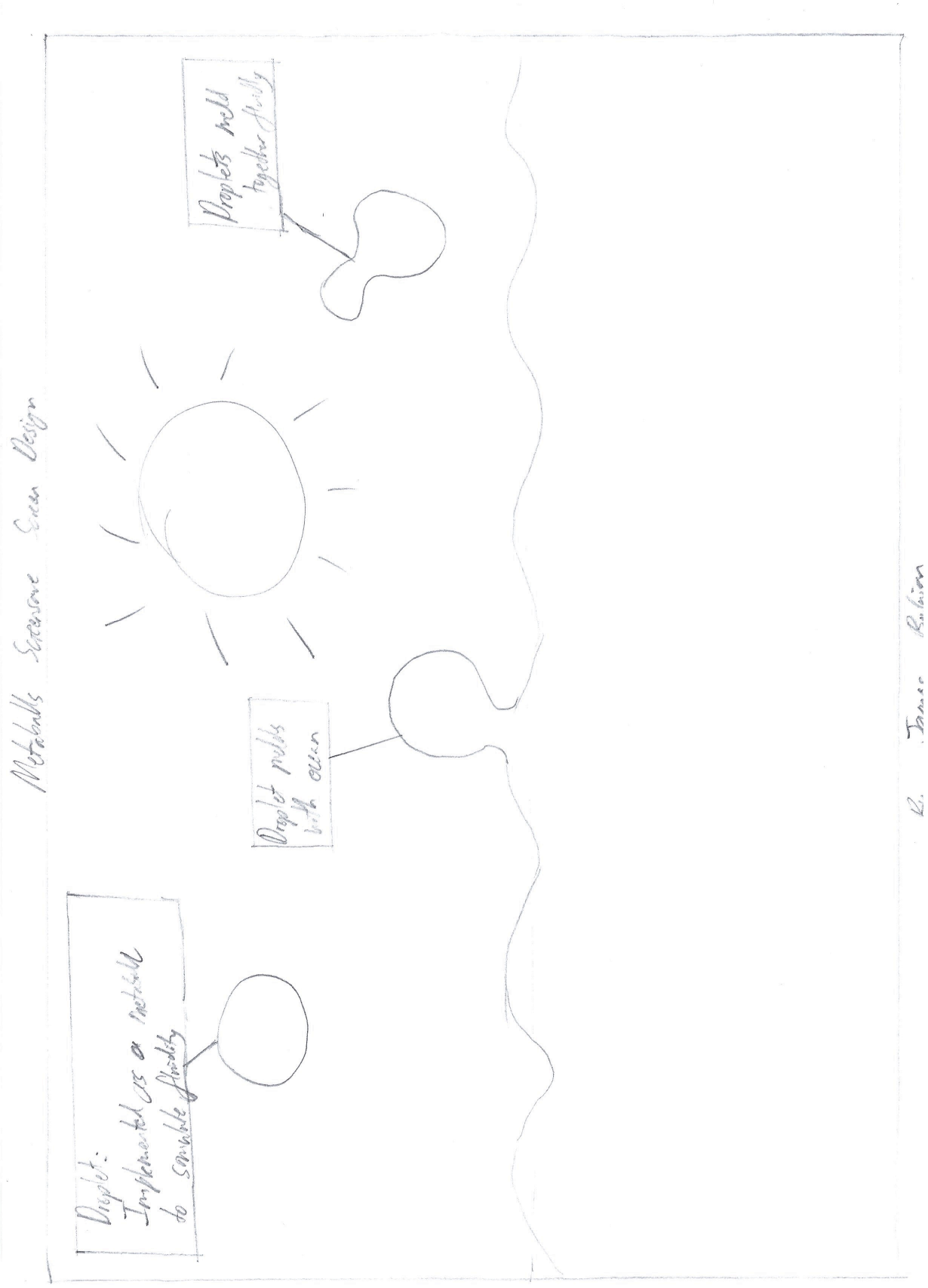
8.1.14 Clouds and Mountains Screensaver Screen Design



8.1.15 Mandelbrot Screensaver Screen Design



8.1.16 Metaballs Screensaver Screen Design



8.1.17 Data Dictionary

Clouds and Mountains Screensaver Data Dictionary

Terrain.vb - Language VB.NET

Variable	Data Type	Definition
ridgedMultiFractal	RidgedMulti	Base terrain heightmap noise generator.
amplitudeAdjustedRidgedMulti	Multiply	Multiplies base terrain heightmap by the amplitude of the terrain.
terrainNoise	NoiseMap	Where terrain heightmap is stored
terrainNoiseBuilder	PlaneNoiseMapBuilder	Builds terrain using noise generator and stores it in terrainNoise
terrainNoiseWidth	Integer	Width of terrain noise heightmap
terrainNoiseLength	Integer	Length of terrain noise heightmap
terrainAmpl	Single	Amplitude of terrain heightmap
terrainSmplDist	Single	Distance between each vertex sampled from the heightmap. Determines size of each terrain polygon.
terrainPosition	Vector3	Position of terrain model in world space.
terrainModel	RawModel	Stores the IDs of the VBOs and VAOs of the stored terrain vertex/polygon data in OpenGL for easy drawing.
loader	Loader	Wrapper class to abstract creation of some OpenGL objects.
terrainObjectComponent	TerrainObjectComponent	Handles drawing of terrain objects (trees).
heightMapMask	Bitmap	Image which masks the generated terrain heightmap to ensure some patterns always appear.

TerrainComponent.vb - Language VB.NET

Variable	Data Type	Definition
camera	Camera	Stores the camera's position and orientation as well as its view and projection matrices.
loader	Loader	Abstracts creation of OpenGL objects
terrain	Terrain	The terrain
earth	EarthManager	Stores properties of earth like radius.
sun	SunManager	Stores properties of sun like light direction and position of sun
model	RawModel	Stores terrain model
objectComponent	TerrainObjectComponent	Manages placement of terrain trees
terrainFrameBufferComponent	TerrainFrameBufferComponent	Manages capturing of drawn terrain to a texture so that post processing may be applied to this texture.
shader	Shader	The terrain shader
shadowShader	Shader	The shader used to draw the shadow map for shadow mapping.
shadowBox	ShadowBox	Manages creation of shadow mapping projection and view matrices so that they may cover the camera view frustum.
amplitude	Single	Terrain amplitude
healthyGrassTexture	Integer	ID of healthy grass OpenGL texture
healthyGrassNormalTexture	Integer	Normal map of healthy grass
grassTexture	Integer	ID of grass OpenGL texture
grassNormalTexture	Integer	Normal map of grass
patchyGrassTexture	Integer	ID of patchy grass OpenGL texture

patchyGrassNormalTexture	Integer	Normal map of patchy grass
rockTexture	Integer	ID of rock OpenGL texture
rnormalTexture	Integer	Normal map of rock
snowTexture	Integer	ID of snow OpenGL texture
snowNormalTexture	Integer	Normal map of snow
depthMapFBO	Integer	ID of shadow mapping frame buffer object
depthMap	Integer	ID of shadow mapping texture
lightProjection	Matrix4	Orthographic projection matrix from sun
lightView	Matrix4	View matrix from sun
terrainModel	Matrix4	Terrain model matrix
quadRenderer	ScreenQuadRenderer	Draws quad enveloping screen. Used to blit/draw the terrain texture to the screen.
textureBlitterShader	Shader	Shader used to blit the terrain texture to the screen
terrainResolutionWidth	Integer	Width in pixels of the terrain texture.
terrainResolutionHeight	Integer	Height in pixels of the terrain texture.
fogFalloff	Single	Value used to determine how far until fog begins to set in.
SHADOW_WIDTH	Integer	10000
SHADOW_HEIGHT	Integer	10000

TerrainFrameBufferComponent.vb - Language VB.NET

Variable	Data Type	Definition
currentFrameTex	Integer	Texture holding the current frame
occlusionTex	Integer	Texture storing whether the terrain occludes the view or not

TerrainObjectComponent.vb - Language VB.NET

Variable	Data Type	Definition
objectShader	Shader	Shader for drawing objects
loader	Loader	Encapsulates creation of OpenGL objects
objects	List	List of the objects to be drawn
models	Model	List of possible models to draw
terrAmplitude	Single	Terrain amplitude
terrModelMatrix	Matrix4	Model matrix of terrain
sun	SunManager	Stores sun properties
random	Random	Random number generator
camera	Camera	Stores camera properties
sortedObjectsByDistanceFromCam	Boolean	States if all the objects had been sorted by distance from camera to prevent drawing from overlapping. Set to false when a new object is added to the list, as the list must be resorted. Once sorted set to true.
modelScaleFactors	Double	Size multipliers for each model

ShadowBox.vb - Language VB.NET

Variable	Data Type	Definition
OFFSET	Single	Offset value used in shadow box calculation
UP	Vector4	Vector which denotes which direction is up
FORWARD	Vector4	Vector which denotes which direction is forward

SHADOW_DISTANCE	Single	Distance where shadows are calculated
mixX	Single	Minimum frustum vertex X pos
maxX	Single	Maximum frustum vertex X pos
minY	Single	Minimum frustum vertex Y pos
maxY	Single	Maximum frustum vertex Y pos
minZ	Single	Minimum frustum vertex Z pos
maxZ	Single	Maximum frustum vertex Z pos
lightViewMatrix	Matrix4	View matrix of light from which shadows are casted
cam	Camera	The scene camera
nearWidth	Single	Width of the near plane of the camera perspective frustum
nearHeight	Single	Height of the near plane of the camera perspective frustum
farWidth	Single	Width of the far plane of the camera perspective frustum
farHeight	Single	Height of the far plane of the camera perspective frustum

VolumetricCloudsFramebuffer.vb - Language VB.NET

Variable	Data Type	Definition
currentFrameTex	Integer	Texture to current drawn frame to be post processed.
lastFrameTex	Integer	Texture to last drawn frame without post processing. Used in temporal reprojection as an optimization.
alphanessTex	Integer	An occlusion texture for the clouds. Shows where the background is occluded by the clouds for screen space god rays to be applied.

VolumetricComponent.vb - Language VB.NET

Variable	Data Type	Definition
volumetricShader	Shader	Shader for drawing of volumetric clouds
postProcessClouds	Shader	Shader to post process clouds
quadRenderer	ScreenQuadRenderer	Draws quad to screen
camera	Camera	Stores camera info
earth	EarthManager	Stores earth parameters
sun	SunManager	Stores sun parameters
temporalProjection	VolumetricCloudsFrameBuffer	Stores lastFrameTex and other relevant info used to undertake the temporal reprojection optimization.
oldViewProjection	Matrix4	Stores previous view projection matrix to be used for temporal reprojection
frameIter	Integer	Current frame index, used to sample the bayer matrix for temporal reprojection.
perlinWorleyNoiseGen	NoiseGenerator3D	Generates perlin worley noise for cloud volumetric raymarching
worleyNoiseGen	NoiseGenerator3D	Generates worley noise for cloud volumetric raymarching
weatherNoiseGen	NoiseGenerator2D	Generates the weather map, which defines where clouds appear
curlNoiseGen	NoiseGenerator2D	Generates curl noise for adding wisps in the clouds
perlinWorleyNoise	Integer	The perlin worley noise texture
worleyNoise	Integer	The worley noise texture
weatherNoise	Integer	The weather map/noise
curlNoise	Integer	The curl noise
cloudsResolutionWidth	Integer	Width of resolution clouds are drawn to the screen at

cloudsResolutionHeight	Integer	Height of resolution clouds are drawn to the screen at

GodRaysComponent.vb - Language VB.NET

Variable	Data Type	Definition
godRaysWidth	Integer	Width of resolution God rays are drawn at
godRaysHeight	Integer	Height of resolution God rays are drawn at
quadRenderer	ScreenQuadRenderer	Draws quad to screen
godRaysShader	Shader	Shader to draw screen space God rays
sun	SunManager	Holds parameters of sun
godRaysFrameBufferComponent	GodRaysFrameBufferComponent	Framebuffer to which God rays are drawn to.
camera	Camera	Stores parameters of camera

FrameBufferComponentBase.vb - Language VB.NET

Variable	Data Type	Definition
resWidth	Integer	Width in pixels of texture to which framebuffer draws to
resHeight	Integer	Height in pixels of texture to which framebuffer draws to
prevViewportDimensions	Integer(4)	Stores the results of an opengl query for what the previous dimensions of the viewport were, before the viewport dimensions are changed for the framebuffer. Used to revert changes to the viewport.

Mesh.vb - Language VB.NET

Variable	Data Type	Definition
vertices	List(Of Vertex)	List of vertices of mesh
indices	List(Of Integer)	Indices to the vertex buffer storing which vertex is drawn when. Used to save space on memory used for vertex data, so that vertices for each polygon do not need to be repeated.
textures	List(Of Texture)	List of mesh textures
VBO	Integer	OpenGL vertex buffer object of mesh.
EBO	Integer	OpenGL element buffer object of mesh.

Model.vb - Language VB.NET

Variable	Data Type	Definition
texturesLoader	List(Of Texture)	List of the model textures loaded
meshes	List(Of Mesh)	List of the model meshes
directory	String	File path to model
gammaCorrection	Boolean	Is gamma correction enabled
loader	Loader	Abstraction for loading OpenGL objects

ModelInstance.vb - Language VB.NET

Variable	Data Type	Definition
model	Model	Model used to draw the instance
modelMatrix	Matrix4	The model matrix of the model
position	Vector3	Position of the model

NoiseGeneratorBase.vb - Language VB.NET

Variable	Data Type	Definition
shader	ComputeShader	Shader used to generate the noise on the GPU

RandomFloatGenerator.vb - Language VB.NET

Variable	Data Type	Definition
inst	RandomFloatGenerator	Singular static instance of the random float generator. Implements the singleton design pattern.
random	Random	Random number generator

RawModel.vb - Language VB.NET

Variable	Data Type	Definition
vertexArrayObject	Integer	OpenGL vertex array object of the model
vertexCount	Integer	Number of vertices in the raw model

Shader.vb - Language VB.NET

Variable	Data Type	Definition
vertexShader	Integer	The vertex shader of the shader program
fragmentShader	Integer	The fragment shader of the shader program

ShaderBase.vb - Language VB.NET

Variable	Data Type	Definition
ID	Integer	ID of the shader program

Camera.vb - Language VB.NET

Variable	Data Type	Definition
pos	Vector3	Position of camera
lookAt	Vector3	Position camera looks at
view	Matrix4	Camera view matrix
projection	Matrix4	Camera projection matrix
fieldOfView	Single	Camera field of view
NEAR_PLANE	Single	Distance near plane of camera view frustum is from the camera
scrWidth	Single	Width of screen camera draws to
scrHeight	Single	Height of screen camera draws to

EarthManager.vb - Language VB.NET

Variable	Data Type	Definition
earthRadius	Single	Stores the radius of the earth

HDRComponent.vb - Language VB.NET

Variable	Data Type	Definition

hdrFBO	Integer	OpenGL ID of high dynamic range FBO
rboDepth	Integer	OpenGL ID of RBO
colorBuffer	Integer	OpenGL ID of texture which stores post-processed HDR texture
hdrShader	Shader	The HDR shader
quadRenderer	ScreenQuadRenderer	Renders quad to screen

ScreenQuadRenderer.vb - Language VB.NET

Variable	Data Type	Definition
vao	Integer	OpenGL id of vertex attribute object
quadVbo	Integer	OpenGL id of the vertex buffer object which stores the quad vertex data
loader	Loader	Abstraction for loading / creating OpenGL objects

Screensaver.vb - Language VB.NET

Variable	Data Type	Definition
loader	Loader	Abstraction for loading / creating OpenGL objects
screenQuadRenderer	ScreenQuadRenderer	Used by the entirely shader dependent components. Draws a screen covering quad.
scatteringComponent	ScatteringComponent	Atmospheric scattering
volumetricComponent	VolumetricComponent	Volumetric cloud raymarching
terrainComponent	TerrainComponent	Rasterized randomly generated terrain
godRaysComponent	GodRaysComponent	God / crepuscular rays
hdrComponent	HDRComponent	High dynamic range

postProcessClouds	Shader	Shader which applies post processing to the clouds
camera	Camera	Stores camera properties
sun	SunManager	Stores sun properties
earth	EarthManager	Stores earth properties
time	Double	Internal system clock time.

SunManager.vb - Language VB.NET

Variable	Data Type	Definition
sunPos	Vector3	Position of the sun in the sky

ScatteringComponent.vb - Language VB.NET

Variable	Data Type	Definition
scatteringShader	Shader	Shader for atmospheric scattering
scatteringFrameBufferComponent	ScatteringFrameBufferComponent	Frame buffer for scattering
textureBlitterShader	Shader	Shader for drawing the texture from the FBO to the screen
quadRenderer	ScreenQuadRenderer	Draws quad to screen
camera	Camera	Stores camera properties
earth	EarthManager	Stores earth properties
sun	SunManager	Stores sun properties
scatteringResWidth	Integer	Width of resolution scattering is drawn to the screen at
scatteringResHeight	Integer	Height of resolution scattering is drawn to screen at

BlitTextureToScreen.frag - Language GLSL

Uniform Variable	Data Type	Definition
textureToDraw	sampler2D	Texture to be drawn / blitted to screen

DepthShader.frag - Language GLSL

Uniform Variable	Data Type	Definition
texture_diffuse1	sampler2D	Diffuse texture of model being drawn
texture_emmissive1	sampler2D	Emission texture of model being drawn

DepthShader.vert - Language GLSL

Uniform Variable	Data Type	Definition
lightSpaceProjection	mat4	Projection matrix from light space
lightSpaceView	mat4	View matrix from light space
model	mat4	Model matrix of drawn object

GodRays.frag - Language GLSL

Uniform Variable	Data Type	Definition
lightPositionOnScreen	vec2	Position of light / sun in screen space
occlusionTex	sampler2D	Shows where the background is obstructed from view by the mountains and/or clouds

hdr.frag - Language GLSL

Uniform Variable	Data Type	Definition
hdrBuffer	sampler2D	Texture to have high dynamic range applied to from the frame buffer
godRaysTex	sampler2D	Texture which displays the produced God rays alpha map for blending with the hdrBuffer
exposure	float	Exposure the scene experiences. Used in HDR calculations.
resolution	vec2	Resolution hdr is being drawn at

ObjectShader.frag - Language GLSL

Uniform Variable	Data Type	Definition
texture_diffuse1	sampler2D	Diffuse texture of model being drawn
texture_emmissive1	sampler2D	Emissive texture of model being drawn
sunColor	vec3	Colour of sun
sunDir	vec3	Direction of light from sun

ObjectShader.vert - Language GLSL

Uniform Variable	Data Type	Definition
model	mat4	Model matrix of object being drawn
view	mat4	Camera view matrix
projection	mat4	Camera projection matrix

PostProcessClouds.frag - Language GLSL

Uniform Variable	Data Type	Definition
textureToDraw	sampler2D	Texture to be drawn / blitted to screen
resolution	vec2	Resolution the clouds have been rendered at

scattering.frag - Language GLSL

Uniform Variable	Data Type	Definition
time	float	System time
resolution	vec2	Resolution atmosphere is rendered at
inverseView	mat4	Inverse camera view matrix
inverseProjection	mat4	Inverse camera projection matrix
sunColor	vec3	Color of sun
sunPos	vec3	Position of sun
EARTH_RADIUS	float	Radius of earth
skyColorTop	vec3	Colour of sky top
skyColorBottom	vec3	Colour of sky bottom

Terrain.frag - Language GLSL

Uniform Variable	Data Type	Definition
sunDir	vec3	Direction of sun’s light
sunColor	vec3	Color of sun
healthyGrassTex	sampler2D	Healthy grass texture

grassTex	sampler2D	Regular grass texture
patchyGrassTex	sampler2D	Patchy grass texture
rockTex	sampler2D	Rock texture
snowTex	sampler2D	Snow texture
healthyGrassNormalTex	sampler2D	Bump map of healthy grass
grassNormalTex	sampler2D	Bump map of regular grass
patchyGrassNormalTex	sampler2D	Bump map of patchy grass
snowNormalTex	sampler2D	Bump map of snow
rockNormalMap	sampler2D	Bump map of rock
shadowMap	sampler2D	Shadow map of scene
snowHeight	float	Height at which mountains begin to be snowy
grassCoverage	float	Factor which determines what angle at which grass will cover the mountain, and at which rock will.
fogColor	vec3	Colour of fog
fogFalloff	float	Factor which determines at what distance the fog will begin to onset
cameraPos	vec3	Position of camera

Terrain.vert - Language GLSL

Uniform Variable	Data Type	Definition
model	mat4	Model matrix of terrain
projectionMatrix	mat4	Camera projection matrix
viewMatrix	mat4	Camera view matrix
lightSpaceProjection	mat4	Light projection matrix. Used in shadow mapping for determining the current fragment's position in relation to the

		shadow map.
lightSpaceView	mat4	Light view matrix. Used in shadow mapping for determining the current fragment's position in relation to the shadow map.

volumetric.frag - Language GLSL

Uniform Variable	Data Type	Definition
lightColor	vec3	Sun light's colour
sunDir	vec3	Direction of sunlight
EARTH_RADIUS	float	Radius of earth
time	float	System time
resolution	vec2	Resolution clouds are being rendered at
cameraPos	vec3	Position of camera
inverseView	mat4	Inverse camera view matrix
inverseProjection	mat4	Inverse camera projection matrix
inverseViewProjection	mat4	Inverse camera combined view projection matrix
oldViewProjection	mat4	Last frame's combined view projection matrix
cloudNoise	sampler3D	Perlin-worley noise texture which is raymarched to give the clouds' characteristic fluffy shape
worleyNoise	sampler3D	Worley noise texture is used to create pockets in the clouds
weatherTexture	sampler2D	Texture determines cloud types and positions in the sky
curlNoise	sampler2D	Curl noise texture to give the clouds their characteristic whisps

lastFrame	sampler2D	Previous frame used for temporal reprojection
lastFrameAlphaness	sampler2D	Previous frame alphaness used for temporal reprojection
terrainOcclusion	sampler2D	Used for excluding fragments which won't be seem from being calculated as an optimization
background	sampler2D	Background texture for blending with clouds
frameIter	int	Frame iteration counter for temporal reprojection
CLOUDS_AMBIENT_COLOR_TOP	vec3	Ambient color of tops of clouds
CLOUDS_AMBIENT_COLOR_BOTTOM	vec3	Ambient color of bottoms of clouds

Mandelbrot Screensaver Data Dictionary

Screensaver.vb - Language VB.NET

Variable	Data Type	Definition
time	Double	System time
mandelbrotShader	Shader	Shader for drawing mandelbrot fractal
screenQuadRenderer	ScreenQuadRenderer	Draws quad to screen
zoomPointSeed	Integer	Randomises the zoom point the screensaver starts on
NUM_FRACTAL_ZOOM_POINTS	Integer	Number of different points on fractal to zoom into

Mandelbrot.frag - Language GLSL

Uniform Variable	Data Type	Definition
iResolution	vec3	Resolution screensaver is rendered at
iTime	float	System time
zoomPointSeed	int	Randomises the zoom point the screensaver starts on
selectedPalette	int	Palette user selected to draw mandelbrot fractal in

Shader.vb - Language VB.NET

Variable	Data Type	Definition
vertexShader	Integer	The vertex shader of the shader program
fragmentShader	Integer	The fragment shader of the shader program

ShaderBase.vb - Language VB.NET

Variable	Data Type	Definition
ID	Integer	ID of the shader program

ScreenQuadRenderer.vb - Language VB.NET

Variable	Data Type	Definition
vao	Integer	OpenGL id of vertex attribute object

quadVbo	Integer	OpenGL id of the vertex buffer object which stores the quad vertex data
loader	Loader	Abstraction for loading / creating OpenGL objects

Note: Shader.vb, ShaderBase.vb & ScreenQuadRenderer.vb are the same as in the clouds screensaver

Metaballs Screensaver Data Dictionary

metaballs.frag - Language GLSL

Uniform Variable	Data Type	Definition
iResolution	vec3	Resolution screensaver is rendered at
iTime	float	System time
waveHeight	float	Amplitude of the waves
dropletCenters	Array of vec3	Stores center of each droplet / metaball which falls down
dropletRadii	Array of float	Stores radius of each droplet / metaball which falls down

Screensaver.vb - Language VB.NET

Variable	Data Type	Definition
time	Double	System time
waveHeight	Single	Height of waves
metaballsShader	Shader	Shader for metaballs
screenQuadRenderer	ScreenQuadRenderer	Draws quad to screen

dropletManager	DropletManager	Stores the details of every droplet and applies physics to them

DropletManager.vb - Language VB.NET

Variable	Data Type	Definition
droplets	List(Of Droplet)	Stores information about every droplet
GRAVITY	Single	The gravitational acceleration applied to every droplet
prevUpdateTime	Single	System time at previous physics update. Used to determine time difference since previous physics update to perform physics calculations.
numberOfDroplets	Integer	Number of droplets
spawnBoxWidth	Single	Width of the box in which droplets may spawn
spawnBoxLength	Single	Length of the box in which droplets may spawn
spawnBoxHeight	Single	Height of the box in which droplets may spawn
spawnBoxCentre	Vector3	Position of the spawn box

Shader.vb - Language VB.NET

Variable	Data Type	Definition
vertexShader	Integer	The vertex shader of the shader program
fragmentShader	Integer	The fragment shader of the shader program

ShaderBase.vb - Language VB.NET

Variable	Data Type	Definition
ID	Integer	ID of the shader program

ScreenQuadRenderer.vb - Language VB.NET

Variable	Data Type	Definition
vao	Integer	OpenGL id of vertex attribute object
quadVbo	Integer	OpenGL id of the vertex buffer object which stores the quad vertex data
loader	Loader	Abstraction for loading / creating OpenGL objects

RandomFloatGenerator.vb - Language VB.NET

Variable	Data Type	Definition
inst	RandomFloatGenerator	Singular static instance of the random float generator. Implements the singleton design pattern.
random	Random	Random number generator

Note: RandomFloatGenerator.vb, ScreenQuadRenderer.vb, Shader.vb & ShaderBase.vb are the same as in the clouds screensaver

8.2 Survey Responses

8.2.1 Julia Balajan - Survey Response

SDD Major Project Survey for James Balajan's Screensaver Suite

Name: Julia Balajan

Date: 5/5/19

Q1. Would you use the clouds and mountains screensaver on your computer?

(Yes / No)

Q2. What are some concerns that you have with the screensaver in its current state or some features that you would like to see added?

No concerns
Perhaps, making colours of the grass & sky a bit brighter would make it more appealing

Q3. Would you use the mandelbrot fractal screensaver on your computer?

(Yes / No)

Q4. What are some concerns that you have with the screensaver in its current state or some features that you would like to see added?

No concerns

Q5. Would you use the metaballs screensaver on your computer?

(Yes / No)

Q6. What are some concerns that you have with the screensaver in its current state or some features that you would like to see added?

No concerns

Thank you for taking the time to complete this survey

I verify that I am the one responding to this survey, as listed by the name field above.

Signature: Jo Jup

8.2.2 Ms Saki - Survey Response

SDD Major Project Survey for James Balajan's Screensaver Suite

Name: Ms Saki

Date: 03/05/19

Q1. Would you use the clouds and mountains screensaver on your computer?

(Yes) No)

Q2. What are some concerns that you have with the screensaver in its current state or some features that you would like to see added?

I like the clouds/day/night rising & falling, a slow ~~pitch~~ paced, peaceful music to go with it will make it a complete experience.

Q3. Would you use the mandelbrot fractal screensaver on your computer?

(Yes) No)

Q4. What are some concerns that you have with the screensaver in its current state or some features that you would like to see added?

No concerns - visually stunning & therapeutic.

Q5. Would you use the metaballs screensaver on your computer?

(Yes / No)

Q6. What are some concerns that you have with the screensaver in its current state or some features that you would like to see added?

This one is my favorite & here I would add some enigmatic music.

Thank you for taking the time to complete this survey

I verify that I am the one responding to this survey, as listed by the name field above.

Signature: MSaki

8.2.3 Arjun Malik - Survey Response 1

SDD Major Project Survey for James Balajan's Screensaver Suite

Name: ARJUN MALIK

Date: 12 · 03 · 19

Q1. Would you use the clouds and mountains screensaver on your computer?

(Yes) No)

Q2. What are some concerns that you have with the screensaver in its current state or some features that you would like to see added?

More natural looking clouds;

I verify that I am the one responding to this survey, as listed by the name field above.

Signature: Arjun Malik

8.2.4 Arjun Malik - Survey Response 2

SDD Major Project Survey for James Balajan's Screensaver Suite

Name: ARJUN MALIK

Date: 03.05.19

Q1. Would you use the clouds and mountains screensaver on your computer?

(Yes) / No)

Q2. What are some concerns that you have with the screensaver in its current state or some features that you would like to see added?

It would be nice if in the future a feature was added allowing the adjustment of the rate at which the clouds move.

Q3. Would you use the mandelbrot fractal screensaver on your computer?

(Yes) / No)

Q4. What are some concerns that you have with the screensaver in its current state or some features that you would like to see added?

No concerns, perhaps some feature mentioned above, could be added in the future.

Q5. Would you use the metaballs screensaver on your computer?

(Yes) / No)

Q6. What are some concerns that you have with the screensaver in its current state or some features that you would like to see added?

NA

Thank you for taking the time to complete this survey

I verify that I am the one responding to this survey, as listed by the name field above.

Signature: _____

8.2.5 Samuel Stacy - Survey Response 1

SDD Major Project Survey for James Balajan's Screensaver Suite

Name: Sam Stacy

Date: 13/03/19

Q1. Would you use the clouds and mountains screensaver on your computer?

(Yes / (No))

Q2. What are some concerns that you have with the screensaver in its current state or some features that you would like to see added?

- Clouds are too blocky and seem unrealistic because of a black outline on the clouds.

I verify that I am the one responding to this survey, as listed by the name field above.

Signature: Sam S.

8.2.6 Samuel Stacy - Survey Response 2

SDD Major Project Survey for James Balajan's Screensaver Suite

Name: [redacted] Sam Stacy

Date: Friday - 03/05/19

Q1. Would you use the clouds and mountains screensaver on your computer?

(Yes / No)

Q2. What are some concerns that you have with the screensaver in its current state or some features that you would like to see added?

- Shadow movement not smooth → makes the screen saver seem laggy
- Would be cool if the skyline changed colour, as if the sun was setting

Q3. Would you use the mandelbrot fractal screensaver on your computer?

(Yes / No) → Just not my style / personal preference

Q4. What are some concerns that you have with the screensaver in its current state or some features that you would like to see added?

N/A

Q5. Would you use the metaballs screensaver on your computer?

(Yes / No)

Q6. What are some concerns that you have with the screensaver in its current state or some features that you would like to see added?

-Transition from water to sky seems a bit strange / unrealistic

Thank you for taking the time to complete this survey

I verify that I am the one responding to this survey, as listed by the name field above.

Signature: Sam S.

8.2.7 Julian Peen - Survey Response 1

SDD Major Project Survey for James Balajan's Screensaver Suite

Name: Julian Peen

Date: 12·3·19

Q1. Would you use the clouds and mountains screensaver on your computer?

(Yes / No)

Q2. What are some concerns that you have with the screensaver in its current state or some features that you would like to see added?

Improved graphics. I suggest normal mapping the ground, and Removing ~~floating~~ the black borders around the clouds.

I verify that I am the one responding to this survey, as listed by the name field above.

Signature: JPeen

8.2.8 Julian Peen - Survey Response 2

SDD Major Project Survey for James Balajan's Screensaver Suite

Name: Julian Peen

Date: 3/5/19

Q1. Would you use the clouds and mountains screensaver on your computer?

(Yes / No)

Q2. What are some concerns that you have with the screensaver in its current state or some features that you would like to see added?

Better graphics settings / optimization

Potentially a user-defined setting of quality

Better treetextures

Q3. Would you use the mandelbrot fractal screensaver on your computer?

(Yes / No)

Q4. What are some concerns that you have with the screensaver in its current state or some features that you would like to see added?

N/A

Q5. Would you use the metaballs screensaver on your computer?

(Yes / No)

Q6. What are some concerns that you have with the screensaver in its current state or some features that you would like to see added?

Optimization

Thank you for taking the time to complete this survey

I verify that I am the one responding to this survey, as listed by the name field above.

Signature: JReen