Rendering Invisibility

FINAL REPORT FOR CS39440 MAJOR PROJECT

Author: Katherine Rose Farmer (krf@aber.ac.uk)

Supervisor: Dr. Bernie Tiddeman (bpt@aber.ac.uk)

12th April 2015

Version 1.0 (Draft)

This report is submitted as partial fulfilment of a BSc degree in
Computer Graphics, Vision and Games inc. Industrial Placement (G451)

Department of Computer Science

Aberystwyth University

Aberystwyth

Ceredigion

SY23 3DB

Wales, UK

Declaration of originality

In signing below, I confirm that:

- This submission is my own work, except where clearly indicated.
- I understand that there are severe penalties for plagiarism and other unfair practice, which can lead to loss of marks or even the withholding of a degree.
- I have read the sections on unfair practice in the Students' Examinations Handbook and the relevant sections of the current Student Handbook of the Department of Computer Science.
- I understand and agree to abide by the University's regulations governing these issues.

Signature …… (Katherine Rose Farmer)

Date ……

Consent to share this work

In signing below, I hereby agree to this dissertation being made available to other students and academic staff of the Aberystwyth Computer Science Department.

Signature …… (Katherine Rose Farmer)

Date ……

Ethics Form Application Number

The Ethics Form Application Number for this project is: 983.

Student Number 110028424

Acknowledgements

I am grateful to Bernie Tiddeman for being my supervisor and helping me progress through this project. The amount of mathematical and graphical problems that I had seemed insurmountable at first but Bernie was always there to give me a hand.

I'd like to thank Oliver Roe for being a steadfast rock at my side when the stress of this project was getting too much and sometimes broke me down. I am almost certain I would not have made as much progress if I didn't have such brilliant emotional support and rubber-ducking coming from him! I am extremely grateful to him for being able to provide that while he was also accomplishing his project and can only hope I was as supportive for him,I'd also like to thank many of the other fourth years for being supportive when I was stressed and for being there to bounce ideas off when I was stuck. Special mention needs to go to my fellow Computer Science housemates Gideon Jones and Merddin Emrys who suffered along with me in getting our dissertations done. I'm pretty sure we all deserve a drink at this point! Also a special mention to my other house Craig Andrews for putting up with all the Computer Scientists talking technical jargon over breakfast and for being there at 3am to help motivate to do a little more of my report. Also there to motivate me to take a break, which I need reminding sometimes!

Also to my parents for putting up with all the technical jargon I was throwing their way when they asked how this project was going! They also deserve a mention for raising me, for never asking more of me than I could give, being proud of my accomplishments no matter how small and teaching me the value of hard work and dedication. Most importantly to me, they taught that being poor doesn't stop you from being the best you can be. My father is intelligent, hard-working and kind, regardless of his position as a cleaner. My mother is kind-hearted, ever present in my life and stronger than anyone I know even though she is ill and unable to work. My parents were never able to have the opportunities I have had, but they were the ones who enabled me to have these opportunities. Without them I would not be here, and I can only hope that when they stand at graduation with me, they will understand that part of this is for them, to experience through me what they could not have. I love them both very much.

I'd also like to thank Aberystwyth University and the wonderful, extraordinary lecturers who have for the last 4 years for giving me the chance to complete this course and all the work that has been done to support my continuing progress through this course. This project is the culmination of all that I've learned here and I can only hope it reflects the high quality of teaching I have received here.

Abstract

Invisibility being a realistic possibility, without the use of optical illusions, is a recent discovery and is in large part due to the progress made in the field of transformation optics and metamaterials. With this recent progress, the idea of virtually simulating the theory presented is an interesting one. Hence why this project was undertaken. A virtual simulation can model any potential pitfalls that could occur while science cannot currently create a structure that would render an object invisible to the naked eye using the theories presented.

The main aim of this project was to produce a virtual model that was based on the theory presented and then to adjust any of the parameters involved to see what effects could be achieved. A side aim of the project was also to test the various geometrical shapes the cloak could take and how this would affect model. The project aimed to prove that the theory could be presented in a virtual simulation and that by adjusting certain parameters, certain effects could be achieved.

The model was produced using WebGL in combination with Three.js. The model has been created in a spherical, cylindrical and conical shape, all with a solid and shell structure included. The effects that were achieved were perfect invisibility, a chromatic effect and showing the cloak from a single viewpoint, looking into how the cloak might appear if you were able to view the cloak from one point all the way through. The project has proven that it is possible to render an invisibility cloak using the theory presented and to produce various effects based on the theory. My results are somewhat compromised by the language used and Three.js as the results are inconclusive based on the rendering being controlled by the language in a way that has produced some results that appear to be correct according to the theory but when tested separately produces incorrect results. In conclusion, further work would be needed to create a completely accurate model, mostly into determining how far the language is affecting the model and how that can be counteracted.

Contents

Background, Analysis & Process

Introduction

Invisibility cloaks are starting to enter more and more into the realm of possibility. There are already many projects that both theorise and display practical solutions to the problem of how to create true invisibility such as the 'Rochester Cloak' [1] that using a series of lens of differing focal length to bend light around an object. Currently this is limited by the angles that it can viewed from and some edge problems that ruin the invisibility by hinting at the object cloaked or distorting the background behind the object.

What this project aimed to tackle was the idea of a 3-D rigid structure that would cloak an object within it. The difference was this project aimed to render such a structure in the virtual space and thereby obtain results about how well current theories into metamaterials, transformation optics and their relation to invisibility can be displayed in a virtual space and what this can inform us about those theories. One of the reasons for working in a virtual space when it comes to these theories is that currently they cannot create structures big enough to cloak in all directions and still work on the visible light spectrum in a reasonable amount of time. This is due to the structure needing to be less than a few hundred nanometres for each metamaterial 'atom' in the structure [2]

The direction the project ended up taking was to create multiple geometrical models of the invisibility cloak using the WebGL refraction model and manipulating it to use transformation optics, the mathematical equations behind metamaterials being used to create invisibility cloaks, and then to manipulate those equations to create effects that would be comparable to an invisibility cloak if it were created in reality.

Reason for Project Choice

The author's reasons for this project choice are threefold. One is that the subject of invisibility is universally fascinating and being able to delve more deeply into this project and the subject matter behind was a brilliant opportunity to expand and gather more knowledge in a field they had little knowledge of.

The second reason was that the author's course and skills were more directed towards graphical projects, with previous experience in Javascript, WebGL and HTML5 being most useful for this project. This would provide a good example of their graphical skills as well as an example of their web and research skills. The ability to delve more deeply into extra libraries such as Three.js and to use WebGL for a more complicated challenge was also a draw to accepting and completing this project.

The third reason was that their industrial placement had provided little chance to use their graphical skills in a professional environment or a project of a large size. This meant that will their skills were more directed towards this area, these skills were unused and their other software engineering skills were enhanced. By using this opportunity to enhance their graphical skills while maintaining their other software engineering skills, they will show a wider range of abilities to future employers.

Background

As the author was not familiar with the topics involved with this subject, a lot of background research was required. The research can be broken down easily into the four main topic areas.

Refraction

While it may appear to be a basic area to start with, it was a good place to gain a level of understanding that could be used in the future research topics.

Refraction is the occurrence of light passing between two different mediums and the difference causing the velocity of the light to change. An example of this is the change between air and water. If we look into a pool of water and there is an object at the bottom of the pool, the object will appear bigger than it actually is due to refraction.

Snell's Law is used to describe the relationship between the angles of incidence, the incoming ray of light towards the refractive medium, and the angle of refraction that occurs after the ray has passed into the medium. [3] This law can be used in vector form to obtain the refraction vector using the incident vector and the refractive index of the two mediums the light passes through. The refractive index is a number that describes how light passes through the medium. The formulas for Snell's Law and its vector form can be found in Appendix B.

Another reason for looking into refraction was to understand how the graphical languages being contemplated for this project, OpenGL and WebGL, recreated refraction. This was made simpler because WebGL is based upon OpenGL ES and therefore uses the same implementation of refraction as OpenGL.

The refraction model is based upon the code presented in GPU Gems 2 [4] that describes how to simulate generic refraction. This code is included in Appendix C but will be commented on here. As can be seen from the code, it uses texture mapping to recreate the how refraction should appear in real life. The texture map co-ordinates are transformed by a small amount that is based on the refractive index of the material you are attempting to imitate in your simulation.

This idea is used in OpenGL, but it transforms the texture map using vectors instead. The refract method for OpenGL is also included in Appendix C and be discusses in more detail here. This refract methods uses the vector form of Snell's law found in Appendix B to find the refraction vector. This occurs in the vertex shader, the shader that is used in OpenGL to calculate the vertices of the program and what kind of data is found at each vertex. The refracted vector is then passed to the fragment shader. The fragment shader works out the colour of each vertex so that it can be rendered ass that colour. The easiest way to use the vector in the fragment shader is to have a cubemap that you have passed to the shader available. The vector is then used, much like the NVIDIA refract method, to transformation the map so that it appears refracted according to the refractive index you chose in the vertex shader.

Understanding how this refraction model worked was crucial to implementing this project and therefore implementing the theories that were researched. However it was less crucial to understanding the theory being dealt with in this project. The next few sections will be dealing with the different topics that helped to shape the author's understanding of the invisibility cloaks being theorized and produced in the last couple of decades, due to the relative infancy of the invisibility cloaks created using metamaterials.

Transformation Optics

Transformation optics is a specialized subsection within optics, the study of light. Transformation optics focuses on how to manipulate light through geometric transforms. The way it creates these geometric transforms is via metamaterials.

Transformation optics begins with Fermat's principle. Fermat's principle, also known as the principle of the least time, is the principle that light travelling between two points will take the path of the least time. In Cartesian co-ordinates, this is always a straight line between the two points. Fermat's principle was also a basis for Snell's Law, talked about in the previous section. The reason Fermat's Principle is an important starting point is that Fermat's Principle also works in transformed co-ordinates, the kind that transformation optics deals with. By proving that it continues to work in transformed co-ordinates, and that the shortest path in these co-ordinates tends to be a curved path, proves that by using metamaterials to trick the light into acting as if the space has been transformed, the light can be guided and therefore an object made invisible

However, to work with metamaterials, the light must be treated as a wave, not a ray. This is where Maxwell's equations come in. Maxwell's equations are used to describe the laws that govern electromagnetic fields work. These were the equations that first proved that visible light was on the electromagnetic spectrum by showing the speed of light was the

same as the speed of the propagation of electromagnetic waves. These equations are needed as light must be described as a wave to understand how it would interact with the metamaterials due to the materials they are made of and the way they work. Maxwell's equations are included in Appendix B for reference.

The co-ordinate transformations that are required for invisibility cloaks are such that the co-ordinate leave a hole in space in the middle of the so-ordinates. As such, all objects in that space would be invisible to any waves that are propagated around the space. An example of what this co-ordinate space would look like can be found in [5] in Figure 9.

There is a lot more to transformation optics than what has been discussed here, but these sections should suffice to give an understanding of the kind of research that was undertaken and the topics that are necessary to explain the next two sections.

Metamaterials and Invisibility Cloaks

Metamaterials have been around a lot longer than people may think when they hear the word mentioned. The first occurrence of metamaterials can be found in Roman times. [6] Ruby glass contains nano-scale gold droplets that make the glass appear ruby based on the concentration of the droplets. The difference between this metamaterial and modern metamaterials is the amount of control we have over the structure of the metamaterial.

Two uses of the metamaterials are superlens and cloaking devices, the latter being what this project was about.

Superlens are also an interesting use of metamaterials being used to manipulate electromagnetic waves. A superlens is designed to go beyond the diffraction limit, so there is no loss of resolution at smaller levels. Conventional lens have a limit to what microscopic level they can go too without losing information. A superlens is designed to go beyond this using negative refraction. Negative refraction sounds like something that cannot be, but Sir John Pendry described this effect in a lecture he gave at the Institute of Physics, ICL. [7] [8] By using metamaterials that are constructed to have a refractive index of -1, the lens can go beyond the diffraction limit and super scale imaging can occur. Currently no superlens have been built as the metamaterials required would be non-trivial, but a hyperlens, which puts objects in the near-field into the far field, has been built for the UV spectrum. [9]

For invisibility cloaking using metamaterials there has been many more proof of concepts constructed. Invisibility cloaking uses the theories presented in transformation optics to propagate the electromagnetic waves around a shielded object. A common structure in these metamaterials is a lattice structure made of split rings, described in a paper on the cloak created to work on microwave frequencies [10] The split rings are placed in a lattice formation and work like crystals do upon the waves. These structures have to be smaller than the frequency that the waves you are planning to work with are which means that these structures are likely to be at the nano-level. This is where the problems occur. Creating specific nano structures, without any imperfections, is still very non-trivial. Hence why most invisibility cloaks are proof of concepts or classed as imperfect invisibility cloaks. The invisibility cloaks that this project aimed to simulate were ones found in theory. For a

spherical cloak, the lecture given by Sir John Pendry [7] gives a good example of the kind of structure we're looking at. For cylindrical cloaks, a cloak was presented by K. Elassy et al [11] that describes the fundamentals of designed a cylindrical cloak and includes many examples of the structures involved. These designs will be discussed more in the design section as they are relevant to how the author designed the structures for the virtual representations.

Similar Projects

Another research path was looking into any similar projects to the one proposed. However it appeared after numerous researches that there were no project sufficiently similar to this project that would give be some idea of what systems were available or viable.

The closest project I discovered what a virtual representation of the path of the rays through different geometrical cloaks. This did not deal with a complete virtual representation, but it did include a good system for testing the cloak visually using the ray-traced paths. [12]

There are many examples of invisibility cloaks and cloaking device within video games, such as the tactical cloak from the Mass Effect series.

Example of tactical cloak being used in Mass Effect 3. Image belongs to IGN, Bioware and EA.

While the implementation of these cloaks is nothing close to the implementation of the theories presented, the effects that rendered a cloaked enemy or hero slightly visible were an inspiration for the kind of effects that might be expected from an imperfect cloak on the visible light spectrum. An example of an effect that seems likely to occur is a prismatic or chromatic effect. This is more of an informed assumption that a proven point. Reasons for this are that there is currently not a 3D cloak that works on the visible light spectrum, or research into imperfect invisibility cloaks on the visual spectrum, due to reasons discussed above. However as metamaterials refract light, not dissimilar to crystals, the idea of having an imperfect structure that acts as a prism is not ridiculous.

Analysis

Analysis was not a beginning concern for this project, more of an iterative concern throughout the project. Much of this was to do with continuing research that took place throughout the project and therefore changed what conclusions where previously created through problem analysis. Much of this section will talk about the beginning analyses but

will also dip into the analyses that took place further into the project as these shaped the final product by a large degree. One of the first major decisions was the programming language that was chosen for the project: WebGL.

WebGL is an API for Javascript that is widely supported by modern web browsers such as Google Chrome, Mozilla Firefox and Apple Safari. It is used for rendering 2D and 3D graphics in browsers without the use of a plugin. It is based on OpenGL, which is a multi-language 3D and 2D graphics API that is available to use on multiple platforms including embedded systems. There are multiple reasons as to why WebGL was chosen as the language of choice for this project.

One of the driving factors behind the choice of WebGL was the Three.js library. An early decision in the project was that the creation of the geometrical shape of the cloak, such as a sphere or a cylinder, should be as simple as possible so that the work on the implementation of the invisibility and various effects would not be hindered. Three.js is especially good at dealing in abstractions. Creating a renderer, a scene and objects is extraordinarily easy, compared to base WebGL where it can require a comparatively significant amount of time to create an entire scene and then work on it.

Another benefit to Three.js was that there were numerous examples, thorough documentation and tutorials available which would aid in learning the library as quickly as possible and the possibility that any problems encountered further along in the project would likely have an answer within the widespread Three.js community.

The choice of WebGL was also facilitated by the author's choice of version control in GitHub. The author had used github in numerous projects previously and was aware of its many benefits when it came to using it for version control. Github has extensive documentation on how to use its shell interface and also a desktop version that is simple enough for day-to-day commits. The reason GitHub worked so well with WebGL was that GitHub has support for webpages attached to projects called GitHub Pages. These can be projects by themselves, such as the GitHub Page the author used for the project blog, or webpages attached to already created projects, such as this project. This meant that the GitHub Page could be used as both version control and a display for this work.

The author's experience in using WebGL for a previous university assignment was also a factor in choosing WebGL for this project. As the subject matter was relatively unknown to the author as well as being complex physics that would require some time to research and understand, it made sense to choose a language that was both suited to the project and known to the author so as to reduce any learning time needed for the language and use that time instead for the background reading.

After deciding upon the language, the next decision was what the model would focus upon depicting and the questions that the research and implementation would answer. The main questions that the research was aiming to answer were questions along the lines of:

- **Is it possible to render an invisibility cloak, which functions using the theories presented, in a virtual simulation?**

- Are there any problems with representing an invisibility cloak in a virtual simulation? What causes these problem? Can they be fixed in the virtual simulation?
- Any there are strengths in representing this invisibility cloak in a virtual simulation? Why are these strengths occurring? Can these strength be used when creating the cloak in reality?
- What does the virtual simulation show about the theories presented? Does it confirm them? Does it reveal any new information?
- What effects are created in the virtual simulation, through imperfections or otherwise, that could occur in a cloak produced in reality?

The bold question is the main hypothesis of this project, with the other questions being derived from the main question and also used to obtain more detailed information about the main question. The effects are where a lot of the compromise happened, as well as the type of virtual simulation that ended up being modelled.

An example of these constraints affecting the problem analysis is the difference in what the main concern was at the beginning of the project. It was decided that the structure of the cloak would be a main issue to contend with, as metamaterials are the enabling force behind these types of invisibility cloaks. However being able to create an entirely new structure that interacts with its environment in such a specific way would have required a project all of its own to accomplish. In light of this decision, it was decided to focus on the visuals, with the refraction model that we ended up using being tweaked to work with the theories provided by the research for this project. This may have ended up making it more difficult to manipulate the theory to fit the model, but it meant that what was produced is a good starting virtual simulation for this theory, rather than an example of a metamaterial in a virtual simulation.

As to be discussed in the next section, the lifecycle model decided upon was Feature Driven Development. This meant that a feature list needed to be created as part of the lifecycle. As part of the iterative analysis that was included in the lifecycle, this feature list changed over time. The feature list that was originally decided upon can be seen in the Design Specification (Appendix Something). The later feature list will be discussed in the Implementation section as this is where many of the changes to the feature list occurred. Reasons for these changes were mainly time constraints, such as the change from ray-tracing or photon mapping as a rendering method.

This was a beginning feature as the refraction model in WebGL is more of an approximation than a true-to-life model of it. This meant that it would probably have some effect on the implementation and would probably not produce a realistic result. The ray-tracing and photon mapping would have been a more realistic model and definitely was something the project was aiming towards. The problems with these methods is that they are not particularly viable for real-time rendering due to the slowness of the algorithms. While there are examples of real-time ray tracing in examples such as a demo of refraction in water using ray tracing [13], the algorithms used were extremely complicated to

implement and would have required more time than was available to implement. This meant that the decision to achieve real-time rendering with the WebGL refraction model was more viable, though not as precise.

As can be gathered so far, a lot of this project was unknown at the beginning and unveiled as development and further research became necessary. This is another reason why iterative analysis was required. As some unknowns, such as the complexity of the ray-tracing algorithms and their slow performance, came to light, the project had to take different steps to what was decided upon previously. This was both a boon and detriment to the project. A boon as the project developed in a somewhat natural progression in accordance with the author's skill and knowledge. It was a detriment in that many ideas had to be discarded that could have produced interesting results, such as what the outside world would look like from inside the cloak. While this has been estimated in the finished model, it is not a true representation of the idea of how it could be achieved in a true-to-life model.

Research Method

The lifecycle model that was chosen for this project for Feature Driven Development (FDD). FDD is an agile methodology that was introduced in 1997 by Jeff de Luca and Peter Coad. It is highly adaptive and designed to produce tangible working progress and to track that progress at all stages of the development. It is typically designed for medium to large projects as the code reviews and quality reviews can be hard to achieve as a single developer. However the iterative development and the emphasis on working on features is what drew the author to this particular methodology.

The project contained a great deal of unknowns at the beginning of the project, due entirely to the lack of similar projects and systems available to compare and contrast with and also the complexity of the subject matter. The idea of using FDD was to help deal with these unknowns by structuring the produce into small steps that could be easily adapted depending on how the unknowns affecting the project. By also maintaining quality, as suggest by FDD, the code should be easily refactored for minor changes.

FDD also gave the author the idea of using a project blog to track progress. By updating the project blog and therefore explaining the current features completed and steps taken each week, the project would be traceable and therefore give a good measure of how well the project was progressing. Something that the author also added to track progress and to help with the structuring of the project was to include a design specification and a test specification as deliverable. These documents are normally more associated with the Waterfall model which is not an agile methodology, but plan-driven development. However the author felt that with the amount of unknowns, having deliverables that would enable a structuring of the research and ideas brought forth by that research would help them produce a better quality of project.

FDD have a 5 step model for project development with the last two steps being iterative. These are:

1. Develop an Overall Model

2. Build a features list
3. Plan by feature
4. Design by feature
5. Build by feature

Developing an overall model was included in the analysis stage of this project. By analyzing the project based on the background research undertaken, the author was able to decide on an overall direction the project could take. This was documented in the Outline Project Specification (Appendix Something). Another important section of the project that was developed at this point was what kind of research methods would be appropriate according to the research that was done.

Research methods fall into two categories: qualitative and quantitative. A mixture of these methods is normally required to produce robust results as well as solid proof for the hypotheses presented. Examples of qualitative methods are interviews and observations as these are bettering at finding out the 'whys' and 'hows' of hypotheses and tends to not provide results that must be explained in writing rather than with graphs and statistics. For this project, observational research seems appropriate as it is a visual project, and observations can answer a number of the questions proposed.

Qualitative research methods such as interviews and questionnaires would not be suitable for this type of project as they are more about finding people's opinions and views on a piece of subject matter. If this project was about the ethics of invisibility then these two methods would be suitable to obtain a general sense of how people feel about invisibility in an ethical sense. This is why these two methods have not been considered for this project.

Quantitative research methods were difficult to decide upon as the background research had provided me with a great amount of detail on the theories and mathematical equations needed, but not how those equations might be represented in regards to visible light in terms other than the invisibility of the object contained in the cloak. A paper on invisibility cloaking with regards to ray-tracing [12] did produce a visual representation of how a ray might act within an invisibility cloak. Using this method to produce a similar path that curves around the object would provide a good quantitative measure on proving whether the theory does work in virtual space.

A method that would have been more preferable would have been a table of vectors or indicators of what the differences in vectors might be after they have been transformed by the cloak. This would have been easy to compare and contrast with in regards to the model. However, the background research did not provide such a source and this method was disregarded due to lack of comparison material available.

Another method that was used is using the plotted points on the traced ray and then determining whether the vector direction is correct in regards to what direction the beginning ray was travelling in. This fits in with invisibility requiring the light rays to travel out of the cloak in the same direction as they entered the cloak so they appear un-refracted. However as there are no indications in the papers the author researched that this should

be the case, this could be misleading. This will be taken into account when using the method and then it's weighting in the results.

Design

FDD Design

FDD design is designed to be done by feature. The process is normally such that you will design a feature, then implement it, and the cycle continues as such until all features are complete. This is the point when a project is considered feature complete.

The author adapted this process to include a complete design specification before implementation began. This was used to tie up all the author's research on the project into a cohesive document that could be used as a basis for the rest of the project. This Design Specification can be found in Appendix E. This was especially useful for tracking progress as the initial feature list as well as the initial gathering of information into one document helped to gather the author's thoughts and therefore continue on the project with less unknowns available to them.

The feature list included in the Design Specification was changed as the project progressed. This was mainly due to time constraints. The final feature list, after changes made due to the path of the project took, is included here:

1. Preliminary Research

    a. Research into current theories surrounding invisibility cloaking, metamaterials, transformation optics and refraction.

2. Preliminary Learning of New Technologies.

    a. This included Three.js, WebGL, and Ray-tracing (to determine constraints).

3. Outline Project Specification

    a. This outlines the projects and any deliverables that are required from the project.

4. Prototype of Sphere Shell Structure

    a. Implementation of a spherical transparent shell structure with differing refraction ratios.

6. Design Specification.

    a. Discussion of feature list, model design and system design.

7. First Model – Spherical Invisibility Cloak (Solid and Shell Models)

    a. First fully implemented model created based on design specification and early prototyping.

    b. Includes the effects that were implemented. (Examples include chromatic effect and the multiple viewpoints).

8. Test Specification

    a. Test specification detailing evaluations and tests to be used on current and future models.

9. Second Model – Cylindrical Invisibility Cloak (Solid and Shell Models)

    a. Second fully implemented model created based on design specification and first model.

    b. Includes the effects that were implemented.

9. Third Model – Conical Invisibility Cloak (Solid and Shell Models)

    a. Third fully implemented model created based on design specification and first model.

    b. Includes the effects that were implemented.

10. Viewpoint from within cloaks.

    a. The ability to look from inside the cloak and what kind of images would be shown.

13. Project Diary

    a. A blog that has tracked the progress of the project.

14. Final Report

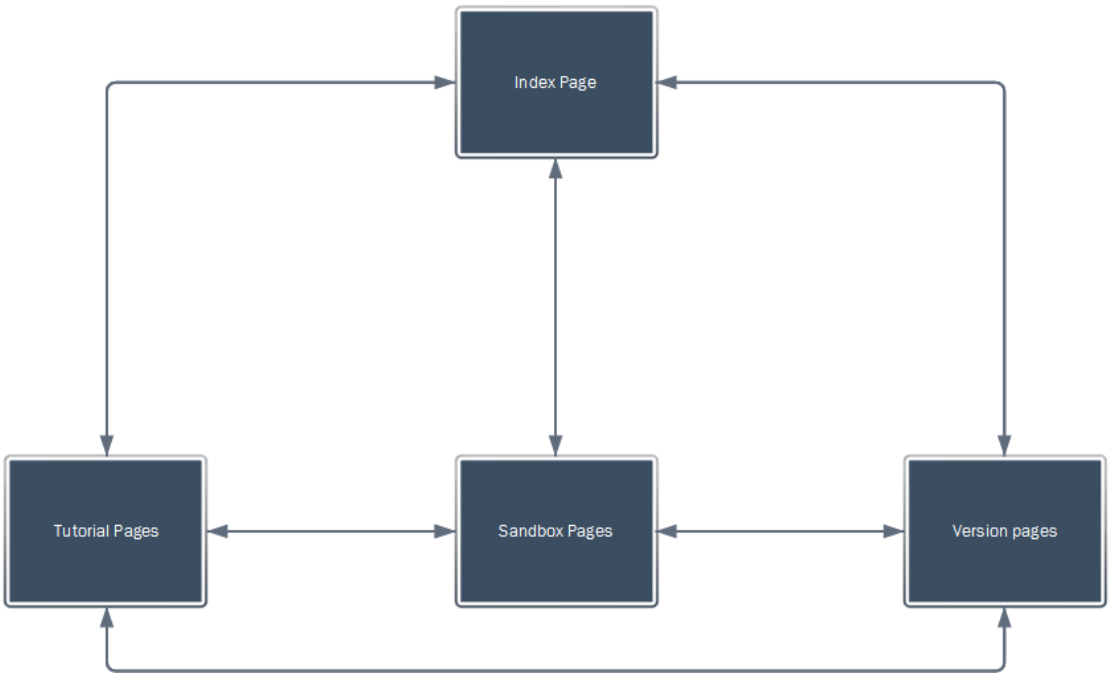
    a. Report detailing project progression, completion and results.

As you can see from the differences between this list and the list in the design specification, there is little to no mention of ray-tracing and photon mapping. After initial attempts at prototyping a ray-tracing renderer, the decision was made that the progress being made on it was too slow for the time remaining. Instead, the models were designed using the Three.js refraction model as a base.

The next two sections describe the designs from both the specification view and the FDD designs that were made as the project progressed.

System Design

The system designed was intended to be a simple hierarchy of web pages. It was designed this way as the web page design was not as important as the working models. The author had considered using a design pattern for the website and the project but after looking in the Design Patterns book by the Gang of Four[14], there were no design patterns that fitted the simplicity of the system. The webpages are designed to follow the MVC pattern and reflect that in their creation. The model is the JavaScript files that run the program, the view is the webpage displayed on the monitor and the controls are defined by a third-party library called OrbitControls.js, mentioned in Appendix A. The hierarchy of the webpages was designed as such:



This system is meant to enable easy access to any point on the site using links/navigation menu provided. The different sections are as follows:

Index Page

The main page of the site that is the first page the user will see. The user will also be able to access documentation from the project from the index page.

Tutorial Pages

This is where the tutorial work from the preliminary research and preparation will go. This will not be tested and is used to show what the initial research consisted of.

Sandbox Pages

This is where any sandbox work for the different versions will be stored. This will include non-final versions of the models chosen.

Version Pages

This is where each version that is a final prototype of a feature will be stored. This will include a first version that shows a spherical object made of glass as a geometrical model prototype, a second version showing the shell structure of the cloak and a third version which will be the final demo version of each of the models.

Model Designs

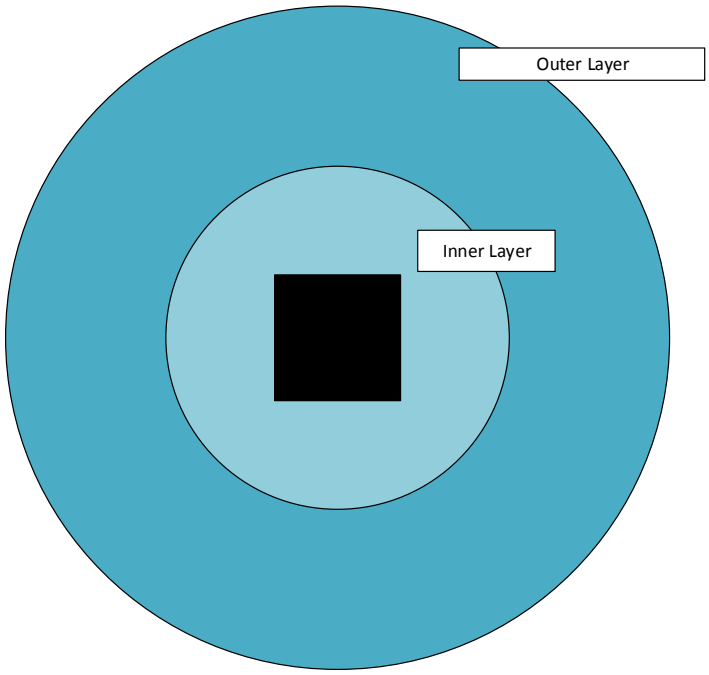
The model designs discussed here will not include any details on the implementation involved and will discuss more about the research into what structures have been theorized for each of the geometric models. For the effects, it will be a more general discussion about the kind of effects that were hoped to be implemented. It will be mentioned in the next section if they were or were not able to be implemented.

For each model, there is two types of model to discuss. The two types are a model with a shell structure and a model with a solid structure. A shell structure expects more than two layers to the cloak and that the light will be guided using the layered design. The solid structure is to have a solid geometrical shape with a hollow centre. It works on the layered basis as well but expects that it will not be created in shells but as a solid single shell. They are not expected to work any differently, as the equations involved will be the same, but the structures are expected to appear differently, with the solid structure only showing certain points of the ray passing through, whereas the shell structure is expected to depict each stage.

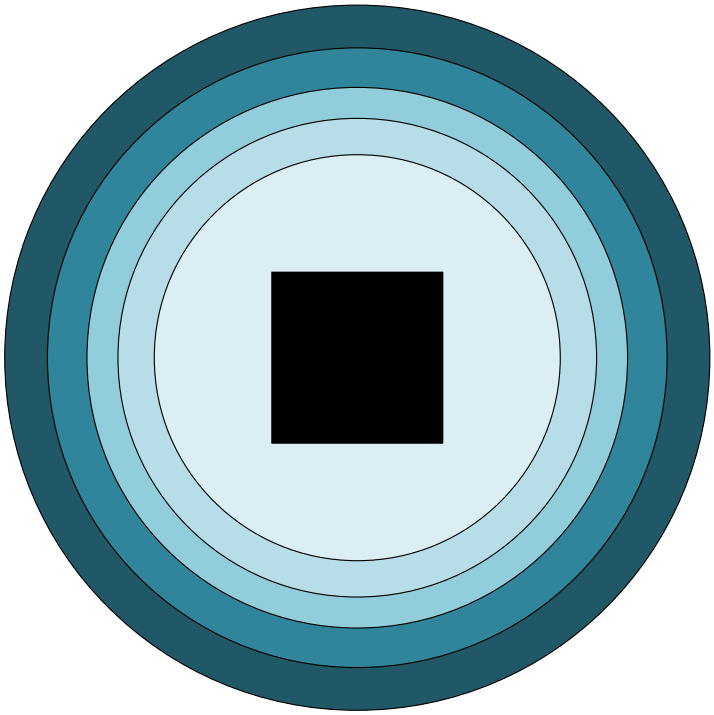
Spherical Model

The lecture that Sir John Pendry contained a diagram of a spherical shell that directed the rays around an inner shell. This is what gave the author the idea of creating a more solid structure that would, in the visualization, contain an outer and inner sphere surrounding an object. This was also the inspiration for having a solid structure model for each geometrical model. The reason for this is that a solid sphere with a hollow core would not allow for traversal through the sphere that would allow you to observe the different shells from inside.

The design for the solid sphere was this:



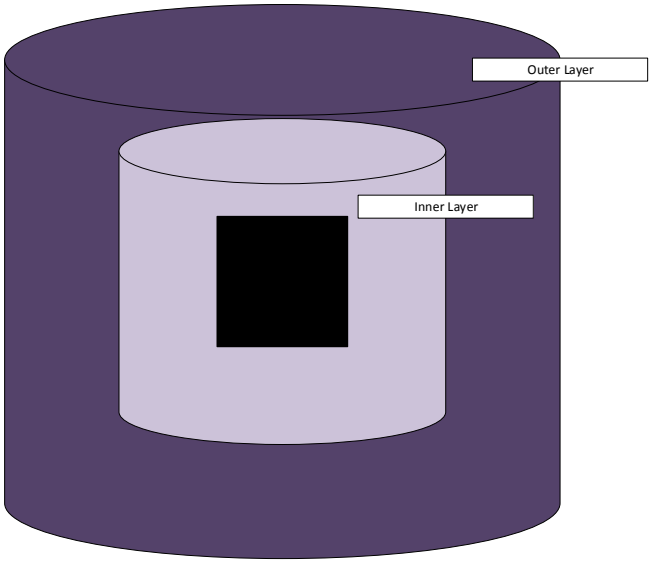
The design for the shell sphere was this:



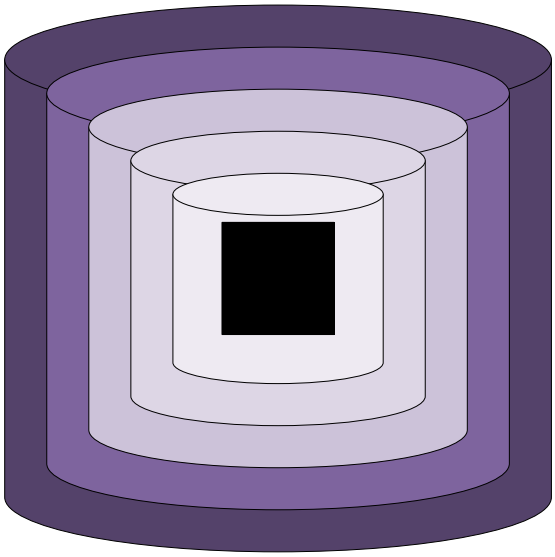
Cylindrical Model

The cylindrical model is the most widely described. The cloak that was designed to work on radio waves [10] was a short cylinder only designed to work on the horizontal plane for the object. The paper which inspired the model the most was one describing the structure of a 3D cylindrical cloak made using silver-silica based metamaterials[11]. It contains a diagram showing how the layered structure would need to be arranged, which contained a shell structure with different refractive indexes. The author decided this would be a useful structure to work with and might work for other models as well.

The design for the solid cylinder was this:



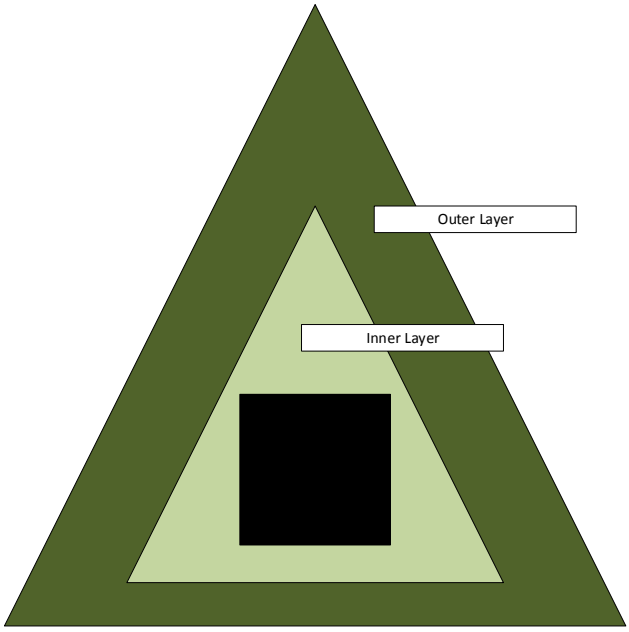
The design for the shell cylinder was this:



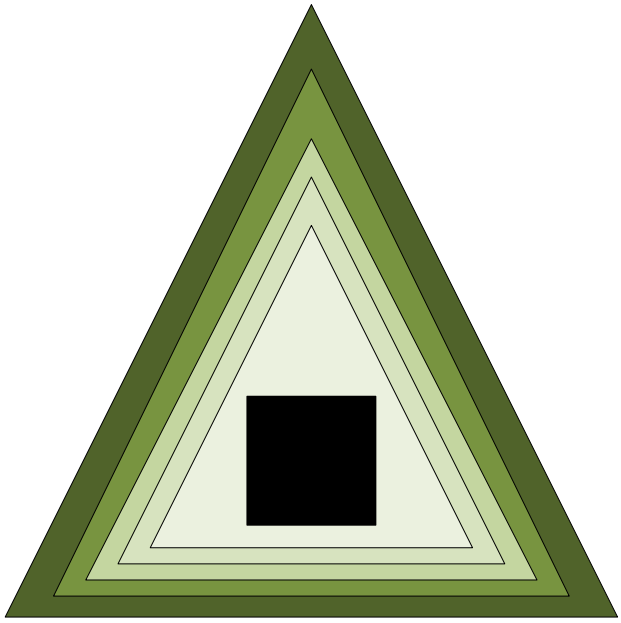
Conical Model

The idea for the conical cloak came from the paper that was the closest to this project. It discussed invisibility cloaking using ray tracing and had virtual models of many different structure. While the author had already found enough material on the spherical and cylindrical models, it did contain examples of ellipsoidal and conical models. The conical model appeared the most interesting out of the two, the top point being an interesting factor when considering manipulating light.

The design for the solid cone was:



The design for the shell cone was:



Effects

The effects that were discussed with the author's supervisor were these:

- Chromatic/Prismatic
  - This was theorized to occur when the cloak had some imperfection that caused it to split the light as if it was a prism.
- Caustics
  - This is the name applied to the light scattering that happens on the surface of water when it ripples. If the cloak had had a flexible surface this may have been a viable effect, however the author decided it was not appropriate to the cloaks being designed.
- Multiple/Single Viewpoint(s)
  - The author decided it would be interesting to see what happened if you treated the cloak as if the light was only coming from the front half of the cloak but you could still see the back, how would the visualization handle that? Multiple viewpoints is what happens normally when the light comes from all angles, at this point you would expect invisibility no matter what angle you looked at it from.
- Centre Rays/Inner View
  - The central rays cannot physically travel all the way round in the same amount of time it would take to travel straight through the cloak. This is due to the constant speed of light. The author and their supervisor theorized that if you could use the central ray in some way to add light to the inner part of the cloak, the person inside a cloak might be able to see out of the cloak.

The only effects from these that were decided to be used was Chromatic, Single Viewpoint and Inner View. These were not designed before most of the implementation had occurred, as these relied on the implementation for the cloak being correct. These effects will be discussed more in the implementation section.

Implementation

Tools

The tools used for creating the website and the project were relatively minimal. The version control was GitHub [15] which was mainly accessed via the desktop client, though there were a few cases where the Git shell was necessary as the checkout function was not accessible through the desktop client.

The editor used for the JavaScript files, which contained all the WebGL code for the models, as well as any editing needed for the webpages, to update links between the pages, was done in Atom.io editor[16]. Web development doesn't tend to need an IDE so the author chose to use an editor that had a customizable colour scheme, to highlight the different parts of the files, and was able to work with all the different types of files involved with the project blog and the project webpages. It also was able to connect to the project's GitHub repository and keep track of the changes being made.

There was a small program implemented in Java to test the different mathematical equations before placing them into the shader files. This was because the shader files do not have good debugging tools available, so being able to test any logic failures outside of the main program was useful throughout the project. It was a lot faster than testing the maths by hand and meant that any logical fallacies could be worked out relatively quickly. This was worked on in NetBeans IDE [17], which includes good support for GitHub repositories so any commits could be done in the IDE. It also kept track of any changes and has excellent debugging and code checking tools available.

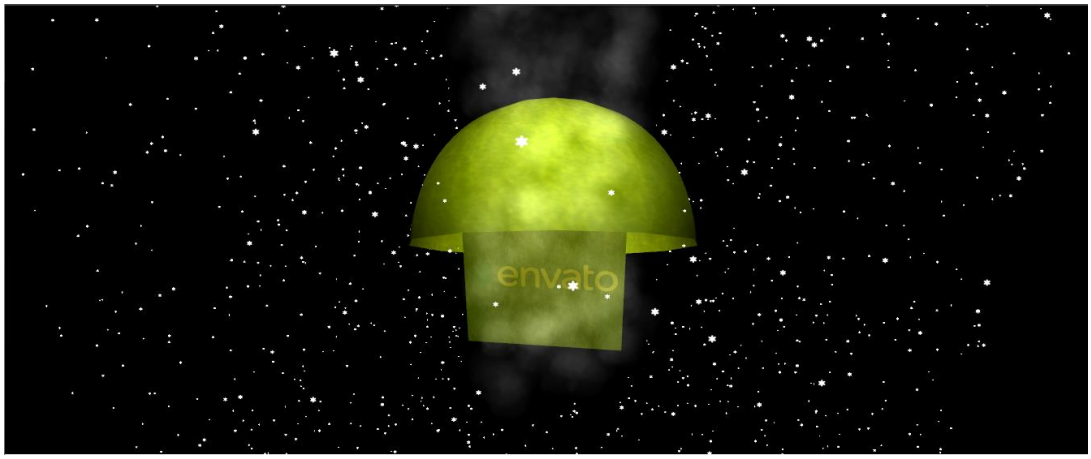
Initial Implementation

The initial coding was based around improving the author's proficiency with WebGL while also learning how to use the three.js library as well as setting up the two websites to be used during the project.

Three.js

The author accomplished learning how to use three.js using a tutorial on Tuts+ [18] that explained how to set up an initial scene and how to use the various geometries associated with three.js.

The tutorial introduced the concepts of Geometries, Materials, Meshes and many other types of three.js objects. The author discovered that three.js was good at abstracting away a lot of the complexities involved in starting a WebGL program. This is definitely something that the author was looking for in using three.js because the program surrounding the model was not as important as the model itself and therefore being able to set up a working scene for the model quickly was essential for getting an early start on implementing the model.



This is the finished result of the tutorial. The different object displayed here are CubeGeometry, SphereGeometry (used to create the hemisphere) and ParticleGeometry (to create the snowstorm).

Three.js also contains many examples on its website detailing how to use the various geometries and a well maintained support form. The author of three.js, mrdoob, also spends a lot of time answering questions on Stack Overflow[19]. Having a good support system in place was definitely something the author needed for this project as there were many unknowns involved in creating the model from scratch and the author anticipated problems occurring.

GitHub Pages

The author also set up the two GitHub Pages that would be used throughout the project. The project blog was set up at krf12.github.io while the project site was set up at krf12.github.io/RenderingInvisibility. The project blog was made using the default theme and was not changed from the original style. The project blog posts were made using markdown, which was easy to use and produced professionally styled blog posts.

The project website was set up using a theme in the GitHub Pages that fit the type of look the author wanted for the project website. The html was then edited to include links to the outline project specification and to the three.js tutorial work that the author was working on. This was relatively simple as the author had previous experience using HTML5. The tutorial page was also edited to include a section that the JavaScript file would be loaded from, creating the canvas area that the model could be viewed in. The same structure was used throughout the project to ensure that the site conformed to the same style. The only pages that would be subject to change were the sandbox pages, which were treated more as test areas and did not include the themes associated with the rest of the site.

Problems with GitHub Pages

The author did discover relatively quickly into implementation, even during the implementation of the tutorial that the GitHub servers seemed to not be able to load WebGL quickly. This is still a continuing problem. This might be fixed with some performance management or using better servers to hold the project, but the creation of the model was deemed more important than performance and thus the GitHub servers are currently sufficient.

Version One

The first version to implement was a structure that would be used for each subsequent model. The easiest way to do this was to find a skybox texture that could be used for the future models and create a glass sphere. These two parts would create a model to build upon.

Skybox

The tutorial had not provided knowledge of how to create a skybox in three.js, so the first step was to look for a tutorial on how to implement a skybox and what kind of texture image was need to create 'realistic' surrounding. This was something that was important to the author as it was a virtual simulation of something that could feasibly work in real life. Using a texture that depicted real-life scenery would enable a more true-to-life test in the future.

The tutorial that the author found was on a blog by romano1la [20]. This blog post explained both how to create a skybox and how to map that onto an object. These two skills were both things that would be required for this first version and could be built on top of for future versions.

To create a skybox, a cubemap texture was needed. Luckily, there is a creative commons site that contains scenery from around the world in cubemap form, designed to be used for skyboxes. The only requirement to use the textures was to include a copy of the license [21] on pages using the photo. The specific skybox used was from Måskonåive [22] and depicted a mountainous scene, which seemed like an interesting background to use for an invisibility cloak. The relatively uniform division between sky and land would make it easier to see any refractions that were occurring wrongly and rectify them.

This texture was then applied to a large cube that served as the boundaries of a scene. The second skill applied more to the next step in the first version.

Glass Spheres

As the spherical cloak was to be the first model on the feature list, this version was used to create a sphere that used refraction to appear like glass. This appeared to be a good test of the refraction model used by WebGL and three.js and is a good test of the knowledge gained from the tutorial.

A glass sphere is created in three.js by created a Mesh that has a SphereGeometry object, which describe the radius and number of segments that make up the surface of the sphere, and a Material object, which will be explained shortly, to describe the sphere. The segments in the geometry decide how well rendered the sphere will appear. Too few segments and the sphere will begin to look more like a multi-faceted object, Too many and you might be using too much power just to render a sphere. The normal amount to make a sphere appear completely spherical with no fragments on the surface is 64.

The Material is what describes how the object looks, whether it looks metallic, reflective etc. The way to create a refractive material that looks like glass is to use the cubemap you created for the skybox and create a refractive map out of it. This can easily be done in

three.js using the CubeMapRefractionMapping method when loading the cubemap as a texture. This mean that when you use the cubemap in a Material object, by adding it as an envmap variable, and you specify that the object is refractive, it will apply that refraction to the map and then map the object with the cubemap that is respective to how refractive the object is described as. This is described using a refractive ratio between 0 and 1.

By combining all this properties, you can easily create a glass sphere, like the ones that were implemented for version one. This version was implemented to look like the solid structure for one half of the future models.

This is the finished result of version one. As you can see, there are two glass spheres, one inside the other. The refraction ratio of the spheres is the same as the refractive index for glass, as you can tell from the upside down image.

Having implemented the refraction in this manner, there were already problems emerging based on what the author's research was suggesting. The refraction index could only between 0 and 1, which did correspond with the refractive indexes shown in the cylindrical cloak structure in [11], but as at this point it was not clear whether or not these were refractive indexes, and therefore yet to have the ratio between them worked out, as this is the number that WebGL uses to work out the refraction, or that these were those values were the ratios between the shells.

If the latter was correct, the ratios were not a problem. If the former was correct, then the refractive method being used here would not be adequate for the model being used. This would not be determined until the sandbox stage, as the next implementation was the shell structure, according to the feature list, not the sandboxing or the theoretical physics.

Version Two

The second version to implement was also the next feature on the feature list to complete; the prototype of the shell structure. The shell/layered structure was proposed in the cylindrical cloak design in [11]. In this version, the problem of how to affect one side of the model over the other was attempted to be tackled. As the refraction needs to change from one half to the other half, if you're looking at it from a single viewpoint and following a single ray or set of rays from a similar direction, there needs to be a way to texture each side differently.

For this version, there was an attempt at creating hemispheres in three.js and then placing them so they would appear spherical. These would then be textured differently. An explanation on why this seemed to be the correct course of action will follow.

Shell Structure

Following on from the last version, the shell structure was created using multiple spheres. In the early version of this, it was made simply using a for statement that increased the shells radius each iteration as well as decreasing the refraction ratio each time to replicate what was expected in the invisibility cloak.

This shell structure was still made of glass spheres and not implementing the theory presented in the papers read during research. However it showed the difference in refraction ratio easily and meant it was easy to see that the iterative method was working. There was however a realization at this point. If a ray was to travel all the way through, the ratio would change halfway. If the project was aiming to show what it might look like if we were able to follow a ray through, then the half way point on each shell should have a different material with a different refraction ratio.

After researching how to fix the problem, the author discovered three ways around the problem. The first way was to apply two materials as suggested by a Stack Overflow answer [23]. However as the answer states, there was some distortion. This would disrupt the look of the invisibility cloak. The second was to use two hemispheres and to place them in the correct position to look like a complete sphere. This will be talked about in the next section. The third way was to use a ShaderMaterial instead and use the surface normals to restrict the textures to the correct halves of the sphere. This ended up being the method used, but was not included in this version. It was used in the sandboxing section after problems arose with the hemispheres.

Hemispheres

As suggested above, the version two ended up using hemispheres to display different refraction ratios on either half of the each shell layer. This seemed like a simple task but ended up being extremely difficult to accomplish in a satisfactory manner.

The first way suggested by another Stack overflow answer [24] and only works well when there is one hemisphere. It presents a hollow hemisphere, and can have another hemisphere of smaller size contained in size and still look good. However, placing another hemisphere on the other side has an odd effect. If you at if from the side you expect to be convex, it is instead concave. Having tried multiple inputs to try to fix this, the author accepted that this method was not going to solve the problem.

The next method was also proposed by a Stack Overflow answer [25] and included a method to create a hemisphere using a LatheGeometry. This involved specify the radius and using the equation proposed by the answerer to create an array of point for the LatheGeometry. This seemed like it would solve the problem at hand and was implemented into the second project. It is still the solution in the current version two but as can be seen from the screenshot below, it does not produce a smooth hemisphere. No

matter how small you make the angle detail value, meant to increase the smoothness of the hemisphere, it still appears more like a multi-faceted geometrical shape.

As can be seen from the screenshot, the sphere is not smooth at all. This produces very poor shells and ruins performance as well.

There is also the other problem with creating the hemispheres this way. The performance drops dramatically, to the point where it cannot even load in a reasonable amount of time on the GitHub page. The author has kept this as a record of the attempts to make a structure this way to show the performance issues and because the third way of creating the hemispheres, with ShaderMaterial, would require a large amount of refactoring, which would be easier to accomplish in the next stage of the project.

Sandboxing

This stage was when the project began to incorporate the mathematical formulas and cloak designs discussed in the research. It is also where the ShaderMaterials began to be used which meant the beginning part of this stage was to refactor the code and create shader files. The shader files were all based on a Fresnel Shader by alteredq, which is included in Appendix A as a code sample, but has been heavily edited in every instance of the models as it was mostly used to ascertain how to write a three.js shader and also to see a way of creating the chromatic effect used to create the bubbles in the Fresnel Shader as this could be useful in recreating such an effect for the invisibility cloak.

Initial Sandboxing

The first thing that needed doing in the sandbox stage was to attempt to create the hemispheres using ShaderMaterials. The author decided to work on the solid structure and used the code for the first version as a basis for the refactoring. The ShaderMaterials require a shader to be written describing how the object is to be rendered. There are a number of standard shaders contained in ShaderUtils.js in the three.js library but the shader for this needed to be something that could be edited by the author as it would also contain the mathematical equations.

This was accomplished by looking at the three.js examples and using the Fresnel shader example to base the shaders for this project on. This particular shader was used as it showed how to use texture mapping from a cubemap in the shaders, what the previous

materials had already been doing to recreate refraction, and also how to use the refract method in GLSL to do this texture mapping so that the material would make the object appear refracted. This was also a good shader to look at as the Fresenel shader was used to create a chromatic effect that made glass spheres appear as bubbles. This effect was similar to the effect that the author wished to recreate for the later effects, though it was particularly muted here and the author wished to change it to be a more extreme effect so as to show a more imperfect invisibility.

Shaders and Models

The solid structure was easy enough to adapt to use the ShaderMaterials, though it was at this point that the author realized that each layer would require a different shader as passing variables to the shader to distinguish between the layers would not work and each layer only appeared as the first layer, therefore the first shader loaded. This was not a problem for the solid structure as it only required two shaders. The shell structure however would require ten shaders which would create a lot of work for refactoring and maintenance. However no solution to this problem seemed available and as such was the same problem at the end of the implementation as well as at this point.

While the shaders had not been completely finished at this point and the feature list said that the next feature was the prototype of the spherical invisibility cloak, the author decided that considering the amount of shader required for each model, it might be easier at this point, while refactoring was a minimum to create other geometrical models. The cylindrical model was the first to be finished and was created using the CylinderGeometry which was very similar to the SphereGeometry but included a height value as well. The conical cloak was not added at this point but created a little later into the project when the author had decided that the conical cloak was going to be the other model. It was both based on the ray-tracing paper's conical cloak[12], as stated in the earlier design section, and that the cone was built using the CylinderGeometry as well, which made refactoring the code to create the cone model was a lot easier as it did not require working out how to use another geometry in three.js, which was particularly time-consuming if the geometry was not well explained in its use, which was a problem that occurred when using the LatheGeometry earlier.

The next step after creating the base models and the shaders needed for the shell and solid structures was to update the refraction to be more realistic and closer to what an invisibility cloak required to be good virtual representation.

Updated Refraction and Refraction.java

It was at this point that the maths in the papers needed to be examined to decide what parts were related to the structure, and therefore beyond the means of this project, and what was required for the refraction. Most of what was required for the refraction was found in the last papers the author looked at, the paper on ray-tracing[12] and the paper on the cylindrical cloak design[11]. The author suspected that the reason for this was that

the metamaterials were the more interesting points in the other papers and therefore were more heavily featured and discussed. In these two papers, while focusing on the cloak designs and therefore structures, it also focused on the fundamentals behind the cloak. This included the path that light would need to travel through the structure.

Before using the maths in the program, the author wanted to ensure that the maths being used would produce results that seemed reasonable. The assumption that the author made was that the vectors would decrease as they moved towards the centre and then increase as they moved away and returned to something similar to the starting vector. This assumption was made because of the structure described in the paper on the cylindrical cloak design[11]. After re-visiting the papers and re-reading them, the author realized that the design of the shells contained details on their refractive indexes, not the ratios between each shell. This was a basis for the above assumptions because, as shown by the paper, the path of the ray would travel through each shell on its path at most twice. The ratios would be reversed half way through and as such the reverse refraction would be applied to the path of the ray. Hence the assumption that the vectors would go from a beginning point, and then back to it, or at least something similar in the same direction.

This was an assumption and the reason for the need for assumptions was because the papers were heavy on the steps taken to reach a final formula, but not the vectors/numbers being inputted and outputted from the formulas. This meant that by using the formulas, the assumption was that whatever came out should be correct and produce the correct effect: invisibility. But as stated, this needed testing to ensure that the maths produced something that seemed usable, so no NaNs or zero vectors. The way the author went about this was by writing a small Java program that would run through the iterations of one path that would go through all shells. This meant ten iterations inwards and ten iterations out. The reason for writing this in Java was the author's experience in using Java. By writing the program in a language that they were highly familiar with, they could write the program in a minimal amount of time and then edit it as new information came up easily. The program contained a third party file, included in Appendix A, that describes a Vector object, not the java.utils.Vector object used by Java, that contains method for dot product, normalizing etc. These methods are extremely useful in the implementation of the maths required.

The formulas used in the implementation are contained in Appendix B as well as Appendix C. The formula in Appendix B is used for the first and final iterations to work out the refraction between the air and the cloak. The code for the refract method used by OpenGL provided that formula for refraction as it was the same as the vector form of Snell's Law, which seemed the most likely formula to use as the ones suggested by the papers required the use of tensors. Tensors required knowledge of the materials being used in the structure which wasn't what the author was aiming for.

The next step after confirming that the maths was working was to add it to the shaders. This was easier for the solid structures due to the smaller amount of shaders required for it. This was another stage where it showed that the amount of shaders in the shell structure was a problem but with how GLSL works, at a very low-level with no easy way to move

variables between different instances of different shaders, this was a problem that was going to be encountered constantly and just meant more time was set aside for any work on the shell structures.

The maths was implemented the same in each shader, using the normal the three.js shaders additions provided. This did present a problem that may have caused inaccuracies in the maths. As the normal were not carried from shader to shader, as there was no way of doing this, the normal vectors were almost definitely calculated as different vectors at each layer. However, as the spheres were centred on the same origin point and only have different radiuses, the author theorized that the normals would have the same direction as the bigger sphere, producing extremely similar results that any inaccuracies might be too small to cause problems.

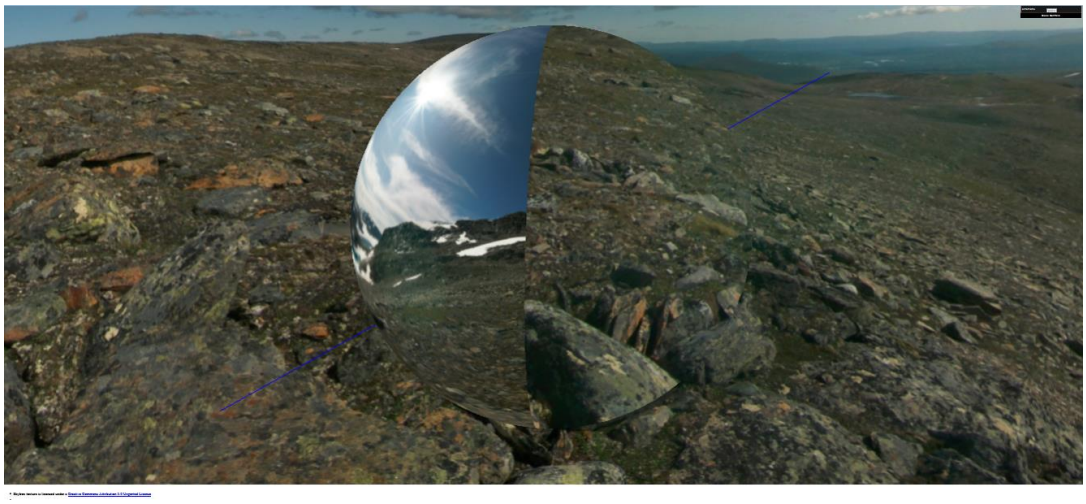
The implementation first only contained the vector form of Snell's Law, applied in code form by OpenGL as mentioned before. This was changed from GLSL to Java quite easily and then placed in an iterative loop. At first the vectors appeared to not be following the expected pattern. This was rectified by using the air to cloak refraction as well, which appeared to fix the problem by making the vectors correct to start with. The vectors used to test this implementation where the position of the light in the scene implementation (1000, 1000, 1000) and a surface normal that had been worked out using the formula found in Appendix B.

After assuring that the vectors seemed to working correctly, the code was placed in the shaders. The author found that it was easier to not use a for-loop as debugging the shaders was not an easy task. The only good way to debug a shader is using the fragment shader to output colours for if statements. If the for-loop had gone wrong, it would be harder to pinpoint the step it had gone wrong at. By spacing the statements out, they could debug any of the vectors to see if any outputted a zero vector. The vectors were then split between the beginning and end vectors for the outer shell and the middle vectors were split between the beginning and end of the middle vectors for the inner shell for the solid structure. For the shell, the vectors were chosen for whatever shell layer they were at currently. This all occurred in the vertex shader.

In the fragment shader, the beginning vectors were applied to the front of the sphere and the end vectors were applied to the back of the sphere. This was accomplished by only applying the end vectors as the fragment colour when the normal z co-ordinates were less than 0. These were then used in conjunction with the cubemap and the texturecube method to work out the color to be used at the current vertex.

At first this seemed to not be working, the front half of the outer sphere was appearing as expected, an invisible sphere. The inner sphere appeared to be a further refraction on the outer sphere, appearing more distorted but maintaining the invisible effect by showing what lay through the sphere, and therefore, what the author expected at that stage. The back half however was appearing extremely solid. It had the convex look to it that showed that the cloak was there. This was extremely troubling and did not appear to have a solution. The effect was improved after removing the normalization of vectors after the first

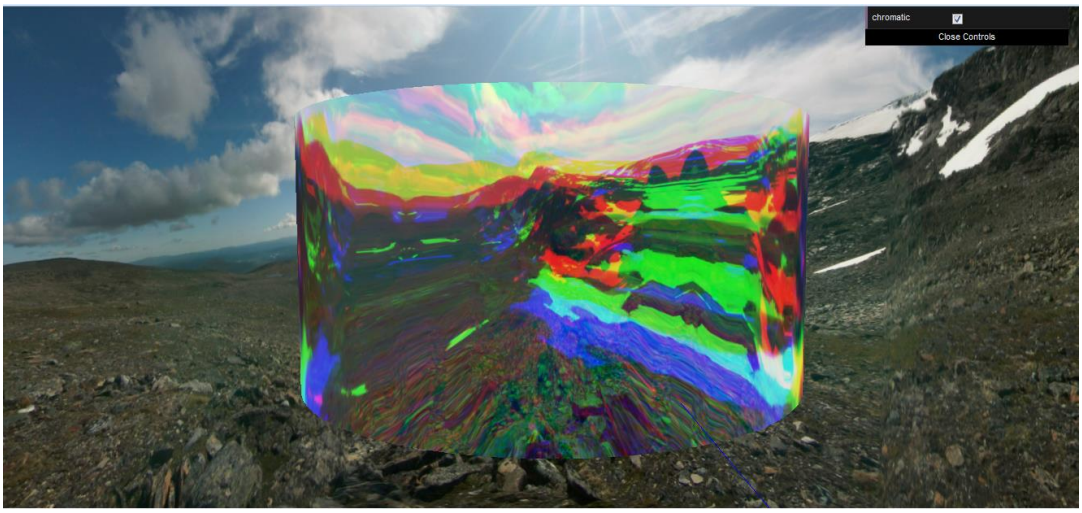
incident vector, as Snell's Law states that the vectors that are produced are normalized refraction vectors when using the vector form of Snell's Law. However it remained convex.



After looking back at the maths and the research that had been done previously, the author came to the conclusion that this was occurring because instead of taking into account all the rays entering the sphere, it was instead following rays coming from the front of the sphere and exiting through the back of the sphere. This was what was causing the strange effect because this would not occur in the actual cloak. The author decided this was a very interesting effect and while it would not be one that would occur in reality, as light would always be travelling from all angles, it was something to consider keeping as an interesting visualization.

This meant that the cloak needed to take into account multiple viewpoints and rays of light. The author decided the best way to visualize this was to only use the beginning refraction vector calculations on the outer sphere. As the rays would be coming in from all directions, this is the maths that would be applied on entrance. This produced the effect of a perfect invisibility cloak that only appeared slightly off. The author suspected this was something that was due to the cloak being rendered on top of the skybox and therefore the renderer had attempted to distinguish it somehow from the rest of the scene. This was something the author had no control over and therefore left it in.

The next step was to add the effect shaders into each model. The single viewpoint shaders were already part of every sandbox model. To add to this, the author implemented the multiple viewpoint shaders, using the method described in the previous paragraph. The chromatics versions of all these shaders were added as well. The chromatic shaders were created by combining the refraction methods in each shader with the Fresnel effect achieved in the shader these were based on. This was achieved by instead of using one of the refraction vectors, three refractions vectors were used and in the fragment shaders these were applied the red, green and blue values of the fragment colour. This was a good visualization of what might occur if the cloak was imperfect and the rays were split through the layer, though as mentioned before in the research and design sections, this was an assumption based on how light can be affected by structures it passes through.



Ray Path-tracing and Options Menu

As can be seen from the above screenshots, the final sandbox models have a thin blue line traversing them as well as an options menu. This were the final two additions to the sandboxing stage and were built upon in the final versions as well.

Options Menu

The options menu was created using a library called dat-gui [26] which can be seen in the top right corner of the screenshots above. Dat-gui is a library that is used in many different three.js demo examples and gave the author the idea of adding a menu that could switch a model between different effects. The sandbox only contains the options for the chromatic shader effects. The final versions contain the chromatic effect and the single/multiple viewpoint effects.

The author attempted to have the option to change model so that the final version would only be a single model that demonstrated the various geometrical shapes and effects. Unfortunately, geometries cannot be completely changed dynamically, you can't change from a SphereGeometry to a CylinderGeometry while a program is running. There was also the problem of swapping from shell to solid cloak as the amount of global variables that would require changing at runtime would be quite large. There was the option of removing the current model and adding the correct one during run-time. However with how slowly it renders on the GitHub Page servers, this would only cause more problems during runtime and ruin the performance more. Hence the amount of multiple versions.

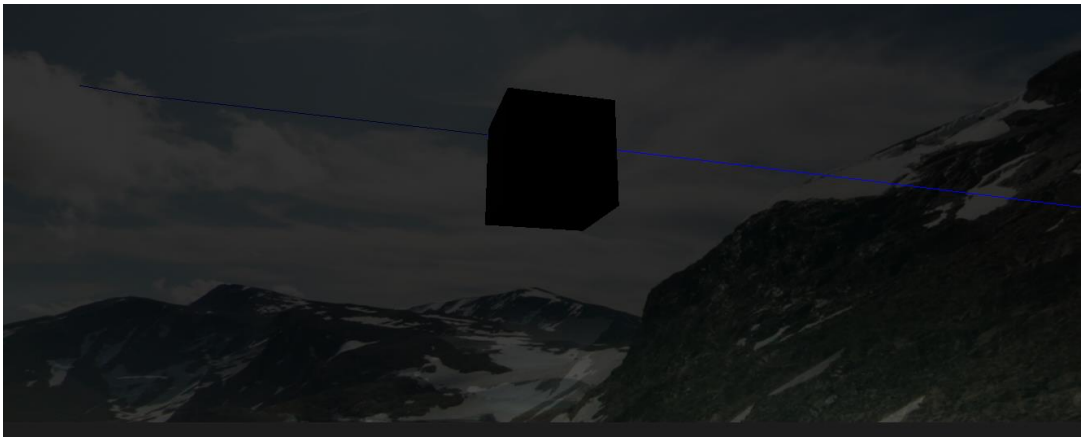
Ray Path-tracing

To accomplish the testing, a ray was needed to be traced using the same maths that was being used in the shaders. With no way of extracting vectors from the shaders, the maths had to be replicated in the javascript files. This meant changing the code from GLSL to JavaScript, which took a little bit of time when learning how to use three.js Vector3 objects. Another change

that needed to happen was to work out how to find a point on the surface of the sphere. This was accomplished using the formula found in Appendix B.

After this, these were then placed in a LineGeometry as the vertices and the line was drawn. At first there was no curve and suspected that the maths was perhaps wrong. After tweaking the maths, changing signs and normalizing vectors, no changes were showing. After further research, the author discovered that to simulate a curve in a line, a Spline is required in three.js. After using this to connect the points in a curve in the right area, which felt like cheating but was unfortunately the only way that three.js allowed the curve to be created from the research done, the line curved in an expected manner.

This forcing of the curve will be taken into account in the testing as it is highly irregular and could be used to disprove the work that has been done. This forcing of the curve also caused the conical cloak to have a line that bent outside of the cloak. It is possible that the reason for this is that while the sphere and cylinder can use the same maths to work out a point on the surface as well as the intersection at each layer, as long as the cylinder takes into account the height as well, the cone needed different maths to work out the intersections and surface point.



Final Versions

These are screenshots of the final versions. The strengths and weaknesses will be discussed more in the testing section of the document. All these screenshots show the models in multiple viewpoint mode and with no chromatic effect.

As can be noted, the shell models appear a lot larger than their solid counterparts. This is a problem that was strange to have occur as the outer shell is the same ratio as the outer shell of the solid model. However the shell appears much larger in comparison than the skybox and barely seems to fit. The author decided to leave the skybox the same size to

show that this was a problem that had occurred and was something that would need more investigation.

Also can be seen that the cone models are not as perfect as the cylindrical or spherical cloaks. This is something that was possibly down to how the cone shapes differs vastly from the other two cloaks. It is likely that in the implementation stage, there was not enough work to take these differences into account and therefore an expectation it would work regardless. As stated above, more about this will be discussed in the testing section.

Spherical Models

Solid Sphere Model

Shell Sphere Model

Cylindrical Models

Solid Cylinder Model

Shell Cylinder Model

Conical Models

Solid Conical Model

Shell Conical Model

Testing and Results

Unit Testing

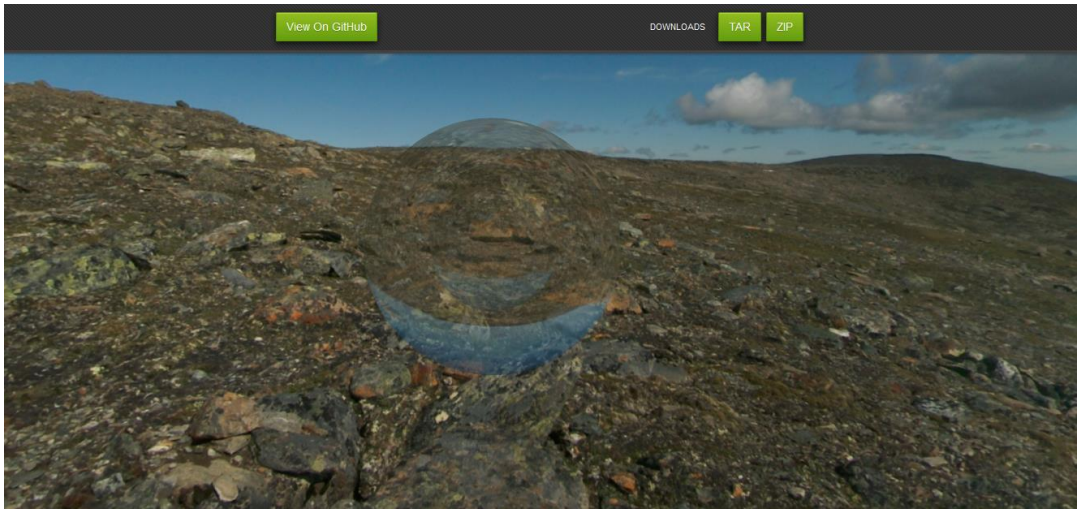
The unit testing will involve ensuring that each file runs and renders as expected. Any issues with rendering will be discussed here if they are relevant to the technical aspects of the project, rather than the experimental aspects of the project.

Tutorial

The tutorial version runs as expected. The particle snowstorm is animated as expected and there are minimal stutters and performance issues. The cube also rotates in the correct manner. As can be see here, all the objects are rendered in the canvas space on the page.

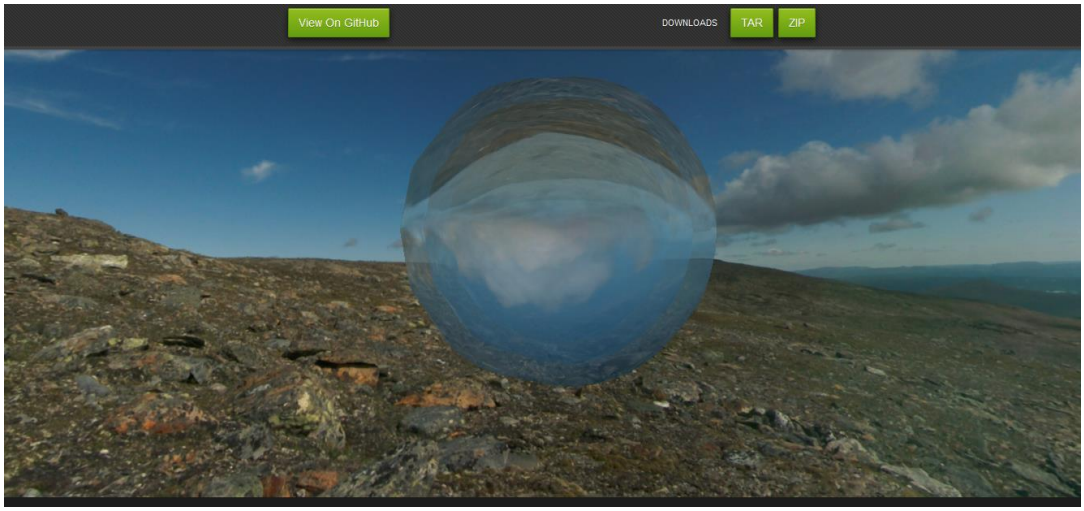
Version One

Version one runs as expected. The two spheres show the correct refraction for glass. The scene rotates and zooms when using the correct motions. There is a small white sphere above the glass spheres indicating where the light is coming from. This is where it is expected and wanted.



Version Two

Version two runs as expected. The different shells show that there is a difference in refractive index between layers via the rendering of the texture cube on the face of each cube. The sphere appears particular low resolution but this is because of the LatheGeometry that was being used to create the hemispheres and is therefore the expected result. It has the same controls as Version One and they still function correctly.



Solid Spherical Model

Single Viewpoint, Invisible

Single Viewpoint, Chromatic

Multiple Viewpoints, Invisible

Multiple Viewpoints, Chromatic

Shell Spherical Model

Single Viewpoint, Invisible

Single Viewpoint, Chromatic

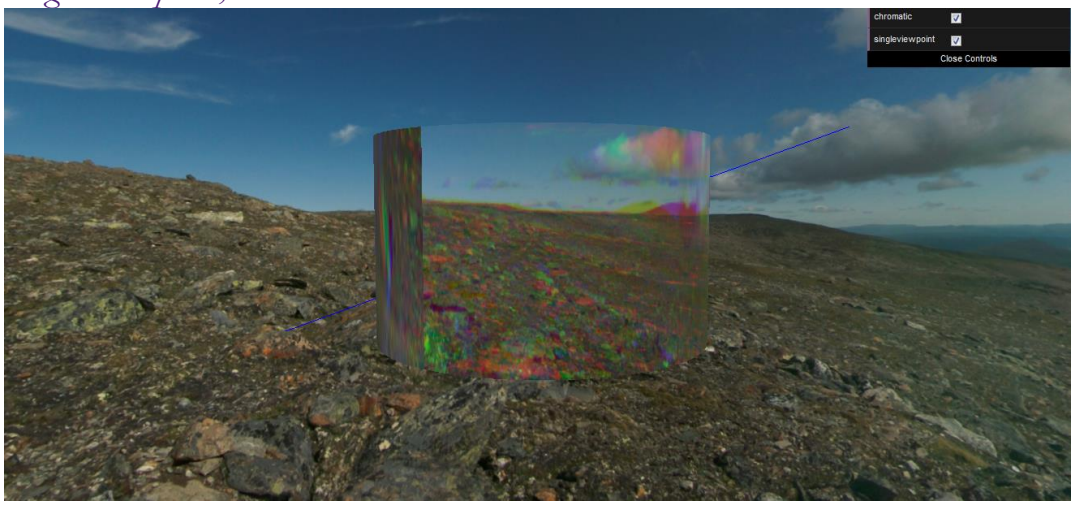
Multiple Viewpoints, Invisible

Multiple Viewpoints, Chromatic

Solid Cylindrical Model

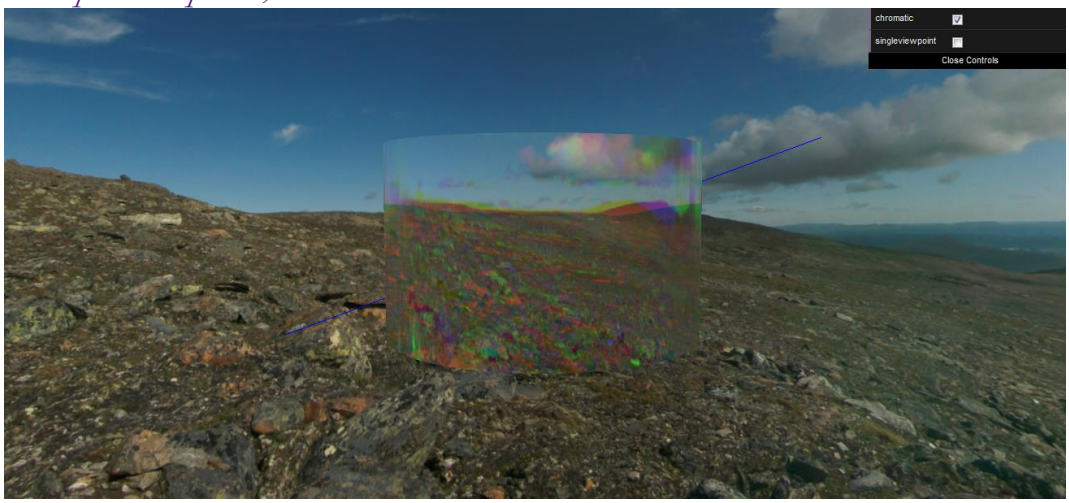
Single Viewpoint, Invisible

Single Viewpoint, Chromatic



Multiple Viewpoints, Invisible

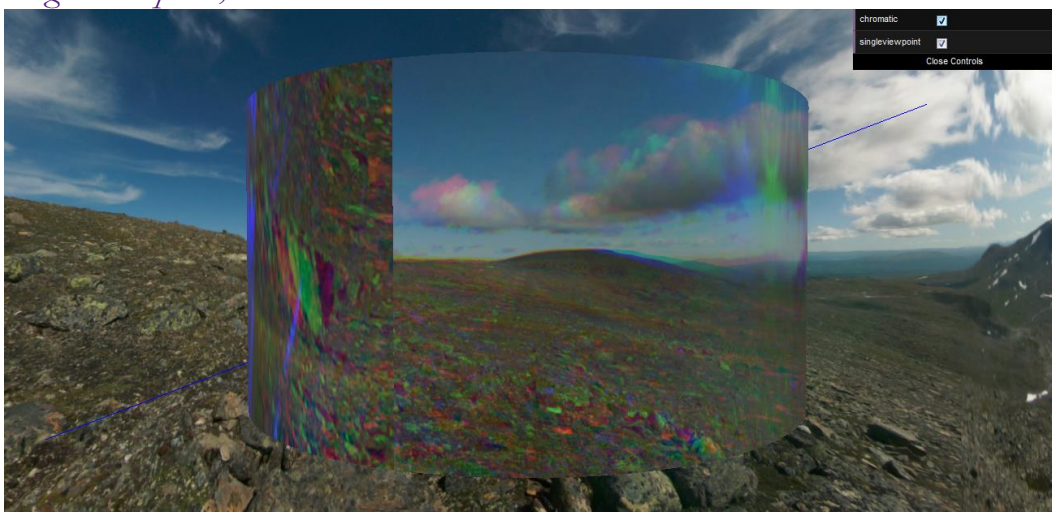
Multiple Viewpoints, Chromatic



Shell Cylindrical Model

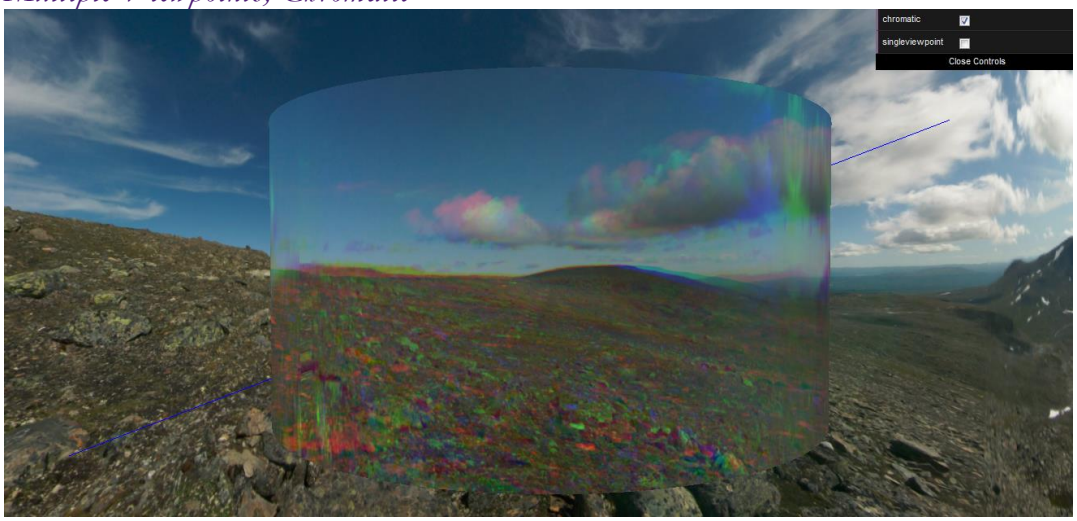
Single Viewpoint, Invisible

Single Viewpoint, Chromatic



Multiple Viewpoints, Invisible

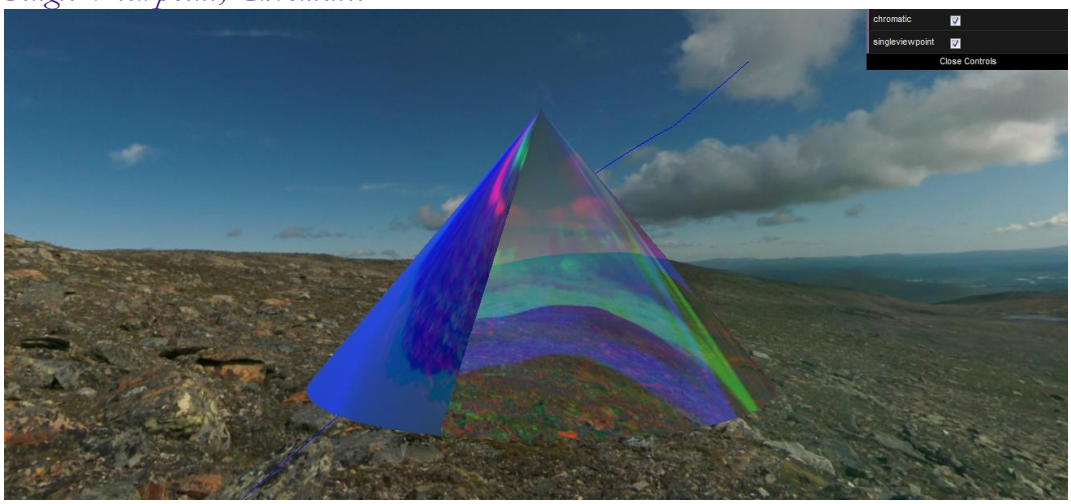
Multiple Viewpoints, Chromatic



Solid Conical Model

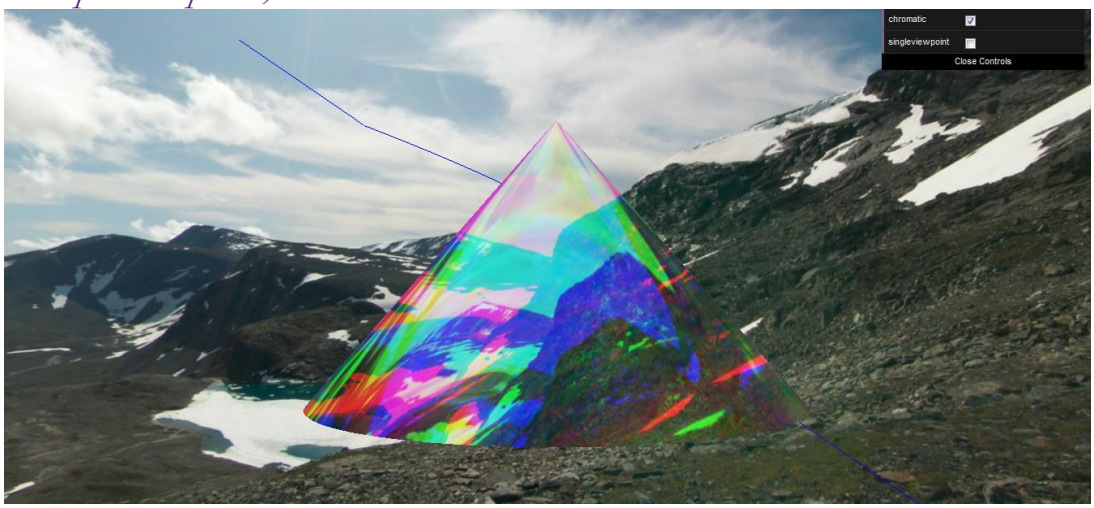
Single Viewpoint, Invisible

Single Viewpoint, Chromatic



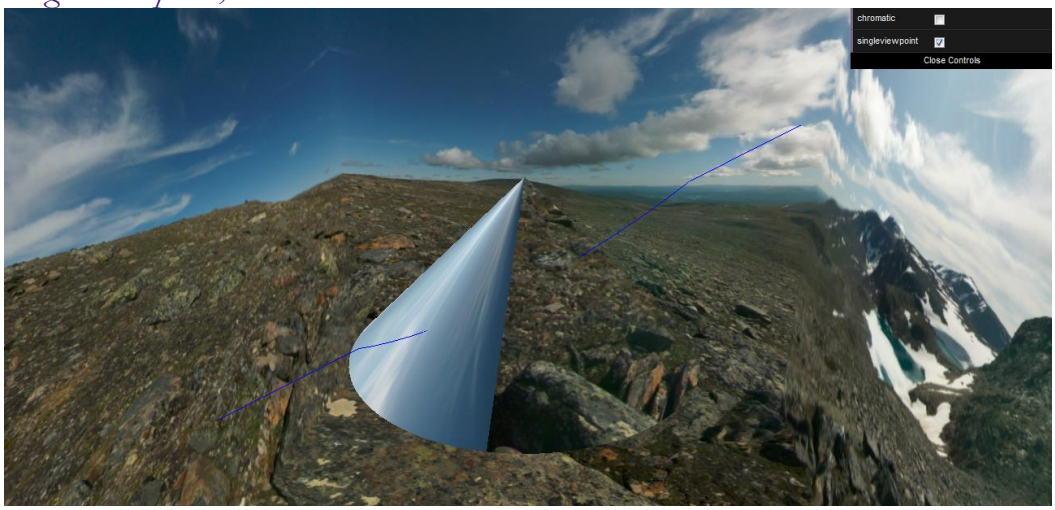
Multiple Viewpoints, Invisible

Multiple Viewpoints, Chromatic

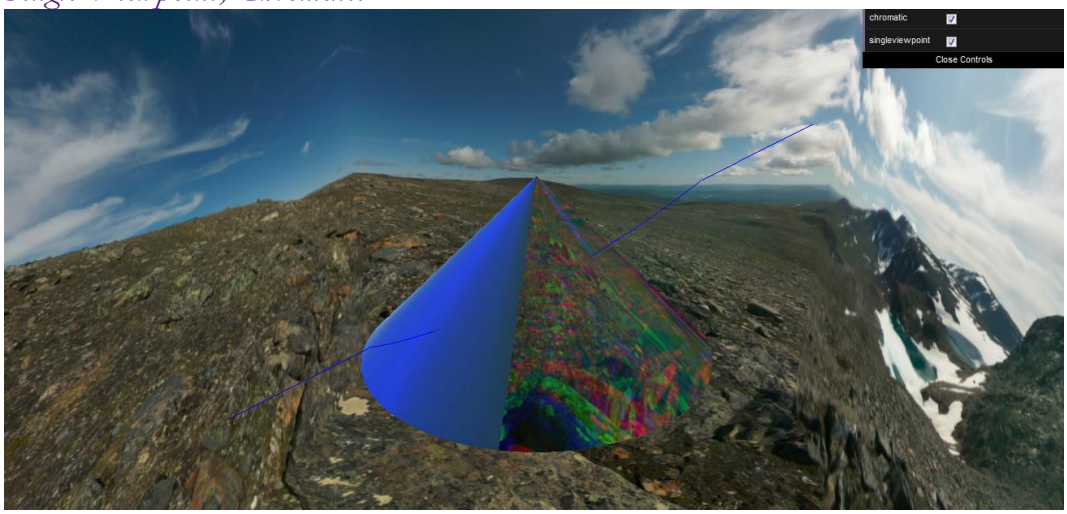


Shell Conical Model

Single Viewpoint, Invisible

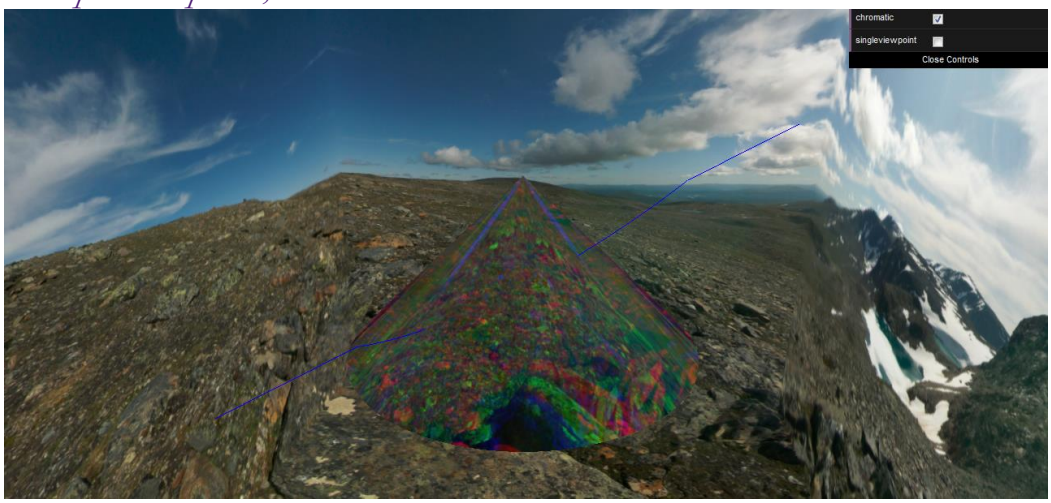


Single Viewpoint, Chromatic



Multiple Viewpoints, Invisible

Multiple Viewpoints, Chromatic



Final Versions

All of the above screenshots, up until Version Two, are the multiple final version models. As there are numerous models with a lot of similar problems, the author decided to place the results of the unit testing for them all together.

The solid structures all function relatively as expected. The halves of the models appear on the correct halves and appear technically functional. The shell structures are were the technical problems occur. All the shell structures should have an outer shell radius that is exactly the same as the outer shell of the solid structure. However, as can be seen from the screenshots, the shell structure appear much larger in comparison to the skybox. It is also very easy when looking at the whole to cloak to see that the cloak is contained in a cube. This suggests that a very small amount of zooming will take the viewer outside of the cube. This is something that is technically wrong and no fix could be found in the time left.

The switching of cameras being outside and inside camera was no problem, though the zooming affects both cameras which is something that would need to be fixed, but lack of time forced the author's hand to move on to other features. The performance is a little slow and sometimes features are difficult to switch between. However considering what is being changed, it is expected that it might be slow to render and some improvement of performance could be done in many places. As this is a proof of concept, the author focused more on getting features working than focusing on them running quickly.

System Testing

The system testing will focus on how the website functioned on a local machine and how it function on the Github Pages server.

Local Machine

The navigation worked as expected on the local machine and took the user between pages as expected. The performance of the pages was relatively good, though the shell structure models were sometimes slow to load. This was probably due to the larger amount of objects that needed loading. There didn't appear to be any system breaking bugs and therefore the system appears structurally sound.

Github Pages

The navigation worked as expected in this area as well. The performances of the pages differed vastly from the performance on the local machine however. Version Two could not load in a reasonable amount of time and the author was never able to confirm if it would ever load. This is possibly due to the LatheGeometry and the multiple shells needed rendering. As it is not one of the final versions, it can pass but it is unsightly having a non-working canvas on one page. The author is not aware of a fix at this stage.

The sandboxes and the final versions were sometimes slow to load but did always load eventually, sometimes it did require re-loading the page when it seemed to timeout but there was never an instance where something as severe as Version Two occurred. This drop in performance could be due to using free servers that have heavy load and therefore are able to render the website quickly. This is something to consider when showing the website to other people.

Browsers

Very late into the project, an update occurred on seemingly all browsers that caused this project to not function correctly. Due to the lateness into the project, the author was only able to determine that Mozilla Firefox at Version 34 and below worked. As of this time, this means that this is the only browser it currently knowingly works on. More testing is required to determine other browser versions but there was very little time left to do this testing at the occurrence of the updates.

Experiments and Results

This section will be split up by the question that the project aimed to answer. Then the experiments used to test these questions will be explained and the results mentioned.

Is it possible to render an invisibility cloak, which functions using the theories presented, in a virtual simulation?

The experiments to test this included

- Observation of the models and their overall effectiveness based on how perfect the invisibility appeared. The multiple viewpoints effect would be tested as this is the closed model to realistic invisibility.
- The vectors that were outputted from the side program, Refraction.java, would be examined to see if they followed a pattern that appeared reasonable according to assumptions made.

As can be seen, these are not foolproof methods but are, unfortunately, the only methods available to the author.

The screenshots in the Unit Testing section will serve as proof of observation with any extra screenshots added if a specific point needs visual proof to be made.

According to these observations, the answer to the above is yes, but there are definitely some caveats. The caveats will be dealt with in the next experiment however, as that is the question being asked there. Here, we will deal with the yes.

As can be seen by all screenshots of the multiple viewpoint models, the invisibility is reasonable perfect. While the models do appear to have an outline, this is most likely produced by shadow effects or by the renderer to show that an object is being rendered. The only observations that do not wholly agree with this are the shell structures, but these is almost definitely because of a technical problem that has occurred rather than a failing in the maths due to the size of the models in relation to their surroundings .

As for the second experiment, the vectors obtained from the Refraction.java program can be found in Appendix G. These are found using the surface normal that is obtained using the surface normal formula for the sphere. As can be seen, the vectors do appear to move around in a circle, which is the assumption that was being made. However the vectors do appear to be extremes. A reason for this is likely to be that the maths is not taking into account the metamaterial's possible structure, as the project was decided to do at the beginning, and therefore the maths might not be as reflective as it could be. So while it fits the assumption, it has some glaring issues.

Are there any problems with representing an invisibility cloak in a virtual simulation? What causes these problem? Can they be fixed in the virtual simulation?

As this questions continues on from the previous question, the same experiments will be used to answer this question. There are definitely some problems with the invisibility cloak in the virtual simulation. The observations show that the conical cloak has some trouble with the ray path-tracing. The curve occurs outside of the cloak and therefore is not reflective of the relatively good visualization. This could easily be because the conical cloak required some different intersection formulas, to determine the normal at the next layer, and by using a generic formula, there were some inaccuracies.

Also, the assumption that doing a virtual simulation without taking into account the structure, though this was done because of the time constraints and the assumption that it would be a large piece of work by itself, could have caused the refraction to not be as true to life as possible. This could be fixed by editing the maths to include anything that is needed and to have the light react to the structure it passes through in a realistic manner. Having ray-tracing would definitely have been as step in the right direction for this. This can be seen causing problems with not only the ray path but in the refraction vectors. The vectors are all extreme in some manner, and this seems unlikely if you are attempting to have a path through a space. Extremes suggests that the vectors are travelling though completely different areas.

Another problem in the shell cloaks. Even though this is a technical problem, you can see that the cloak if not refracting correctly. It is simply overemphasized by the difference in size of the skybox and the model. It can still be seen, on a smaller level, in the solid cloaks. However this is something that may not be fixable in the simulation. As the cloak needs to look as if it hasn't been refracted, and therefore invisible, the only way to have this work in the virtual simulation would be to not refract the object, or render it at all. Even in a position where the invisibility seems perfect, the outline of the model can be seen. This is a definite failing of the virtual simulation.

Any there are strengths in representing this invisibility cloak in a virtual simulation? Why are these strengths occurring? Can these strength be used when creating the cloak in reality?

As this questions continues on from the previous question, the same experiments will be used to answer this question. There are definitely some strengths in representing the invisibility cloak this way. One is that you have no need to create the cloak, which is something that is correctly extraordinarily difficult to achieve. By doing a virtual simulation, you can get an idea of what a cloak might appear like or any kind of problems, or even just a proof that the maths works.

Another strength is that you can achieve something that is close to looking invisible and usable even though it appears that the maths does not have to be entirely accurate or contain any structural details. Though, as said above, this does cause problems, it is helpful that the simulation is not broken and still functions as a worthy virtual example.

There aren't any strengths that could be transferred into a realistic example unfortunately. This could easily be because of the lack of structural knowledge and refraction model that had to be manipulated to achieve these results. However another positive is that effects can be replicated in a virtual environment, effects that might be impossible in the real world, and these can be studied. It would be difficult to create a cloak and then make it imperfect in an arbitrary way. This would be destroying a very expensive piece of gear. Using a virtual environment lets you conjure up new ideas and then apply them without a value piece of equipment being at risk.

What does the virtual simulation show about the theories presented? Does it confirm them? Does it reveal any new information?

The virtual simulation appears to confirm the theories being presented. While this is definitely some differences in the mathematics, the science would appear to be solid and the invisibility cloak being rendered is definitely something that shows the theory off.

However it does not shed any new light on the subject. Much of this is probably that the model was built on the back of the research that was done by many, many talented people. This therefore restricts the new and exciting research to other areas and not what the project was aiming to achieve.

What effects are created in the virtual simulation, through imperfections or otherwise, that could occur in a cloak produced in reality?

The effects that were managed to be created were the chromatic and multiple/single viewpoints effects. The chromatic effect is definitely something that could easily occur in reality without too much effort. It would require an imperfectly structure cloak, perhaps with a defect or specifically designed in such a way, because the light would need to be split rather than diverted. But this effect could realistically occur in reality.

The other effect, the single viewpoint, would not be possible to achieve in reality. It is impossible to be able to see all the way through a cloak that has light restricted to one face. There would still be light travelling through the back of the cloak regardless of the effectors to stop it. It is an effect that was noticed during implementation and was therefore added to show an interesting point that was noticed during implementation, that the author had been able to restrict the refraction in such a way.

Conclusion

Overall Conclusion

This project had shown that the virtual simulation is possible, but there is definitely more work needed to bring it to fruition as a complete virtual simulation. The current simulation works as a proof of concept and can easily be improved to be a decent example of how to create a virtual simulation of an invisibility cloak.

The project has been able to show that while there are a lot of weaknesses in the project currently, the potential is great and therefore shouldn't be ignored as a way of testing out any future additions to the field. Another virtual simulation should also be designed and implemented that takes the metamaterials and the structure required for them into account. This virtual simulation would probably be even more useful to scientists in the field as a resource that could test their theories while the technology to create the structures in reality remains out of reach.

Issues Encountered

Most of the issues encountered in the project were due to the author's lack of knowledge about the physics behind the project and also the lack of understanding of the mathematical formulas that constantly pkagued them. While the subject was fascinating to wade through and delve into, the lack of understanding did at times push the project back and slow progress.

Another issue was having to deal with three.js. While the benefits completely outweighed the problems, the problems could still set back progress. This was most evident when attempting to create hemispheres. This was a lot more work and effort than expected and if the author had been aware of this, they would have moved forward to the sandboxing stage earlier. This is another case of unknowns being a problem, no matter the effort made to uncover the knowledge required to dispel the unknowns.

The final issue was encountering the update that broke the project so late into development. While it was easy to fix when the problem was targeted, by downgrading the version of browser being used, it was a setback for at least two weeks when the problem seemed to be the code being used. It was only after checking through multiple checkout from GitHub that it became obvious it was not the code. Then came the problem of working what the actual problem was. If this had not occurred, there is no doubt more progress would have been made on the project.

Critical Evaluation

Now that the project is finished, this section will serve as a self-evaluation on how I tackled this project, what I would improve and would I have learned from this experience.

Research

While I had done research in a few other modules, this was research on a completely different level to what I had encountered before. There were many subjects involved in this subject that I began with little to no knowledge on and these subjects were complex physics that would most likely cause physicists some trouble explaining. Bernie did invite a physicist to talk me and him about the physics involved but a lot of it did go over my head. This was probably a factor in some of the problems that occurred. I think at this point I have a lot more knowledge about the subject than before and that at this point, I could do a lot more thorough research that I did at the start of the project.

However, based on how little I knew, I feel that the research I did was beneficial to the project. It lead me in the right direction and gave me a lot of different paths to take. In the end choosing to ignore the structure, due to my feelings that the amount of work required to implement it would not fit in the time constraints, may have been wrong and therefore this is something that I would change I was able to do it again.

Design

I stand by my decision to use FDD as the process for the module. I work very well when I have clear, identifiable goals and having the flexibility to both make and change these goals was something that fit both my work ethic and the project's structure. FDD's progress tracking was also something that would give me a trail to follow for this project as well as a way of creating a schedule for myself.

However, I feel that the amount of design I did may have been small in comparison to what was needed. While there was a lot of prototyping, this wasn't as planned as it should have been for FDD. Also, with the deadline looming, the progress tracking began to not be as scheduled as before. If I could do this again, I would work on sticking to the process more. Though I feel the addition of a design specification and test specification was definitely helpful when trying to recall decisions I had made early in the project. I believe that with more progress tracking and more planned design, I might have seen problems sooner or had more time to implement certain optional features if I had worked on designed for them to begin with, such as ray-tracing.

Implementation

I believe that the implementation that was done is well done. It is not the most formal code, but it is robust. I would have liked to take more performance management into account as this became a serious problem towards the end, especially when running the program remotely. This is definitely something I would work on given the chance.

I believe that the time constraints occurring and stopping the implementation of ray-tracing was definitely a loss to the project. While the refraction model in there.js turned

out to be sufficient, the ray-tracing renderer, especially real-time, would have made it a lot easier to implement more realistic refraction. This may have also made it possible to implement other effects or even more specific effects, like splitting a certain ray rather than all.

I also think that choosing to use three.js was most a problem and a benefit. It benefitted me because it made it a lot quicker to set up a program and then from there making changes to that program was a lot quicker because there were a lot less objects to deal with. However this meant that I have less control over some aspect or relied on methods to work as I expected them too. A good example is Vector3 objects and expecting them to not to overwrite the initial vector when using the dot product method. This meant that the implementation had to be more convoluted than necessary to work around the vectors overwriting themselves. I would still use three.js again I think, but I would definitely look more into potential pitfalls and maybe do more than one tutorial as it probably didn't prepare me as much as it possibly could have done.

Testing

I definitely think my testing is not as good as it could have been. A lot of this was the lack of comparative material available to me. I found it incredibly difficult to come up with good tests and test cases when there was nothing to test against. It was all based on assumptions and theories with no visual data. I think that if my research had discovered something I could use as a good comparison, even if it wasn't completely the same that would have been extremely helpful in guiding the testing I needed to do.

Another problem was that having a completely graphical project meant that unit testing and system testing was very arbitrary and could not be automated. Having not had much experience with automated testing past JUnit and some Ruby on Rails experience, I had really hoped to find out more about automated testing during this project. Unfortunately it is difficult to test a graphical program like this in an automated manner.

I think I tackled what I could well however and was able to provide good hypothetical questions as well as answers based on what observations and data I had.

Overall Evaluation

I think that this project has been an extraordinary success and has definitely shown me how easy it is to lose track of a process if you are a solo programmer. Self-policing if definitely harder than I thought it was going to be. I may not have completed it to as high a standard that perhaps me and Bernie expected but what I have produced I am proud of and it has helped me learn to be a better programmer.

I think my main regret is not being able to implement ray tracing and having to work with the refraction model available. This definitely made me feel like I had missed out on accomplishing something incredible. However I think that the visual effects produced and the program working to the degree it does in a short amount of time is amazing and I am proud of myself regardless of the result for achieving this much.

Appendices

A. Third-Party Code and Libraries

Three.js [27]

Three.js is a WebGL library that has been designed to make WebGL easier to use. It reduces the amount of coding that is needed to achiever simple goals, such as creating a cube or a sphere. It is used in this project to make it easier to set up the models and then work on the more complicated parts quickly. No changes were made to the source code. It is open to contribution and bug reporting on Github and is covered by the MIT License[28].

Dat-gui [26]

Dat-gui is a gui designed to be lightweight and change variables in the Javascript code it is attached to. It is used in the project for this very purpose and no changes are made to the source code. It is covered by Apache License 2.0[29].

OrbitControls.js

OrbitContols.js is part of the three.js library and is therefore covered by the same MIT License[28]. It is used to control rotation and zooming of the model and is not altered in any manner.

KeyboardState.js

KeyboardState.js is part of a set of three.js game extensions and was authored by Jerome Etienne. It is covered by the MIT License[28] and has not been altered. It is used to gain access to the input provided by the keyboard in an easy manner. This is then used for switching from the inside to the outside views.

Fresnel Shader

At the start of this project, this shader was not included in the three.js library as standard and was instead included in a three.js example. As of now it is a part of ShaderUtils.js in the three.js library. This code is being included as it was the basis for all the shaders used and while the shaders do not resemble to shader greatly, it is acknowledged as the basis for them and as such should be included. It is covered by the MIT license for three.js [28].

```
/* ------------------------------------------------------------------
//      Fresnel shader
//      - based on Nvidia Cg tutorial
------------------------------------------------------------------ */

'fresnel': {

uniforms: {

"mRefractionRatio": { type: "f", value: 1.02 },
"mFresnelBias": { type: "f", value: 0.1 },
"mFresnelPower": { type: "f", value: 2.0 },
"mFresnelScale": { type: "f", value: 1.0 },
```

```
"tCube": { type: "t", value: null }

},

fragmentShader: [

"uniform samplerCube tCube;",

"varying vec3 vReflect;",
"varying vec3 vRefract[3];",
"varying float vReflectionFactor;",


"void main() {",

"vec4 reflectedColor = textureCube( tCube, vec3( -vReflect.x,          vReflect.yz ) );",
"vec4 refractedColor = vec4( 1.0, 1.0, 1.0, 1.0 );",

"refractedColor.r = textureCube( tCube, vec3( -vRefract[0].x, vRefract[0].yz ) ).r;",
"refractedColor.g = textureCube( tCube, vec3( -vRefract[1].x, vRefract[1].yz ) ).g;",
"refractedColor.b = textureCube( tCube, vec3( -vRefract[2].x, vRefract[2].yz ) ).b;",
"refractedColor.a = 1.0;",

"gl_FragColor = mix( refractedColor, reflectedColor, clamp( vReflectionFactor, 0.0,
1.0 ) );",

"}"

].join("\n"),

vertexShader: [

"uniform float mRefractionRatio;",
"uniform float mFresnelBias;",
"uniform float mFresnelScale;",
"uniform float mFresnelPower;",

"varying vec3 vReflect;",
"varying vec3 vRefract[3];",
"varying float vReflectionFactor;",

"void main() {",

"vec4 mvPosition = modelViewMatrix * vec4( position, 1.0 );",
"vec4 mPosition = modelMatrix * vec4( position, 1.0 );",

"vec3 nWorld = normalize( mat3( modelMatrix[0].xyz, modelMatrix[1].xyz,
modelMatrix[2].xyz ) * normal );",

"vec3 I = mPosition.xyz - cameraPosition;",
```

```
"vReflect = reflect( I, nWorld );",
"vRefract[0] = refract( normalize( I ), nWorld, mRefractionRatio );",
"vRefract[1] = refract( normalize( I ), nWorld, mRefractionRatio * 0.99 );",
"vRefract[2] = refract( normalize( I ), nWorld, mRefractionRatio * 0.98 );",
"vReflectionFactor = mFresnelBias + mFresnelScale * pow( 1.0 + dot( normalize( I ),
nWorld ), mFresnelPower );",

"gl_Position = projectionMatrix * mvPosition;",

"}"

].join("\n")

},
```

Vector.java

Used in the Refraction Java program to implement the vector methods needed to test the maths. A minor change was made in the program to allow for getting out the vector data after it had been added to the Vector, but this is the original code used.

```
/***********************************************************************
******
*  Compilation:  javac Vector.java
*  Execution:    java Vector
*
*  Implementation of a vector of real numbers.
*
*  This class is implemented to be immutable: once the client program
*  initialize a Vector, it cannot change any of its fields
*  (N or data[i]) either directly or indirectly. Immutability is a
*  very desirable feature of a data type.
*
*
*  % java Vector
*  x         =  (1.0, 2.0, 3.0, 4.0)
*  y         =  (5.0, 2.0, 4.0, 1.0)
*  x + y     =  (6.0, 4.0, 7.0, 5.0)
*  10x       =  (10.0, 20.0, 30.0, 40.0)
*  |x|       =  5.477225575051661
*  <x, y>    =  25.0
*  |x - y|   =  5.0990195135927845
*
*  Note that java.util.Vector is an unrelated Java library class.
*
***********************************************************************
*****/
public class Vector {

private final int N;          // length of the vector
private double[] data;        // array of vector's components

// create the zero vector of length N
public Vector(int N) {
this.N = N;
this.data = new double[N];
}
```

```
// create a vector from an array
public Vector(double[] data) {
N = data.length;

// defensive copy so that client can't alter our copy of data[]
this.data = new double[N];
for (int i = 0; i < N; i++)
this.data[i] = data[i];
}

// create a vector from either an array or a vararg list
// this constructor uses Java's vararg syntax to support
// a constructor that takes a variable number of arguments, such as
// Vector x = new Vector(1.0, 2.0, 3.0, 4.0);
// Vector y = new Vector(5.0, 2.0, 4.0, 1.0);
/*
public Vector(double... data) {
N = data.length;

// defensive copy so that client can't alter our copy of data[]
this.data = new double[N];
for (int i = 0; i < N; i++)
this.data[i] = data[i];
}
*/
// return the length of the vector
public int length() {
return N;
}

// return the inner product of this Vector a and b
public double dot(Vector that) {
if (this.N != that.N) throw new RuntimeException("Dimensions don't
agree");
double sum = 0.0;
for (int i = 0; i < N; i++)
sum = sum + (this.data[i] * that.data[i]);
return sum;
}

// return the Euclidean norm of this Vector
public double magnitude() {
return Math.sqrt(this.dot(this));
}

// return the Euclidean distance between this and that
public double distanceTo(Vector that) {
if (this.N != that.N) throw new RuntimeException("Dimensions don't
agree");
return this.minus(that).magnitude();
}

// return this + that
public Vector plus(Vector that) {
if (this.N != that.N) throw new RuntimeException("Dimensions don't
agree");
Vector c = new Vector(N);
for (int i = 0; i < N; i++)
c.data[i] = this.data[i] + that.data[i];
return c;
```

```
}

// return this - that
public Vector minus(Vector that) {
if (this.N != that.N) throw new RuntimeException("Dimensions don't
agree");
Vector c = new Vector(N);
for (int i = 0; i < N; i++)
c.data[i] = this.data[i] - that.data[i];
return c;
}

// return the corresponding coordinate
public double cartesian(int i) {
return data[i];
}

// create and return a new object whose value is (this * factor)
public Vector times(double factor) {
Vector c = new Vector(N);
for (int i = 0; i < N; i++)
c.data[i] = factor * data[i];
return c;
}

// return the corresponding unit vector
public Vector direction() {
if (this.magnitude() == 0.0) throw new RuntimeException("Zero-vector
has no direction");
return this.times(1.0 / this.magnitude());
}

// return a string representation of the vector
public String toString() {
String s = "(";
for (int i = 0; i < N; i++) {
s += data[i];
if (i < N-1) s+= ", ";
}
return s + ")";
}

// test client
public static void main(String[] args) {
double[] xdata = { 1.0, 2.0, 3.0, 4.0 };
double[] ydata = { 5.0, 2.0, 4.0, 1.0 };

Vector x = new Vector(xdata);
Vector y = new Vector(ydata);

System.out.println("x         =  " + x);
System.out.println("y         =  " + y);
System.out.println("x + y     =  " + x.plus(y));
System.out.println("10x       =  " + x.times(10.0));
System.out.println("|x|       =  " + x.magnitude());
System.out.println("<x, y>    =  " + x.dot(y));
System.out.println("|x - y|   =  " + x.minus(y).magnitude());
}
}
```

B. Mathematical Formulas

Snell's Law

$$\frac{\sin\theta_1}{\sin\theta_2} = \frac{v_1}{v_2} = \frac{n_2}{n_1}$$

Source: Wikipedia

Snell's Law – Vector Form

$$r = n_1/n_2$$

$$c = -\mathbf{n}\cdot\mathbf{l}$$

$$\mathbf{v}_{\text{refract}} = r\mathbf{l} + \left(rc - \sqrt{1 - r^2\left(1 - c^2\right)}\right)\mathbf{n}$$

Source: Wikipedia

Fermat's Principle – Optical Path Length

$$S = \int_{\mathbf{A}}^{\mathbf{B}} n\, ds$$

Source: Wikipedia

Maxwell's Equations

Gauss' Law

$$\nabla\cdot\mathbf{E} = \frac{\rho}{\varepsilon_0}$$

Source: Wikipedia

Gauss' Law for Magnetism

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

Source: Wikipedia

Faraday's Law

$$\nabla\times\mathbf{E} = -\frac{\partial\mathbf{B}}{\partial t}$$

Source: Wikipedia

Ampere's Law (with Maxwell's addition)

$$\nabla\times\mathbf{B} = \mu_0\left(\mathbf{J} + \varepsilon_0\frac{\partial\mathbf{E}}{\partial t}\right)$$

Source: Wikipedia

Formula for Air-to-Cloak refraction [12]

$k_2 = k_1 + \lambda n$

$\lambda = -k_1 \cdot n \pm (k_1 \cdot n)^2 + 1 - k^2_1$

k_2 – Refraction Vector
k_1 – Incident Vector
n – Surface Normal

Formula for the Surface Normal of a Sphere

$$x = r \sin\theta \cos\varphi$$
$$y = r \sin\theta \sin\varphi$$
$$z = r \cos\theta$$

Source : Wikipedia

C. Code Samples

Refract Method [4] – Cg

```
half4 main(float2 bumpUV : TEXCOORD0,

float4 screenPos : TEXCOORD1

uniform sampler2D tex0,

uniform sampler2D tex1,

uniform float4 vScale) : COLOR

{

// fetch bump texture

half4 bumpTex=2.0 * tex2D(tex0, bumpUV.xy) - 1.0;


// compute projected texture coordinates

 half2 vProj = (screenPos.xy/screenPos.w);


// fetch refraction map

half4 vRefrA = tex2D(tex1, vProj.xy + bumpTex.xy * vScale.xy);

half4 vRefrB = tex2D(tex1, vProj.xy);


return vRefrB * vRefrA.w + vRefrA * (1 - vRefrA.w);

}
```

Refract Method[30] – GLSL

```
genType refract( genType I, genType N, float eta);

k = 1.0 - eta * eta * (1.0 - dot(N, I) * dot(N, I));

if (k < 0.0)

R = genType(0.0);       // or genDType(0.0)

else

R = eta * I - (eta * dot(N, I) + sqrt(k)) * N;
```

D.Outline Project Specification

Rendering Invisibility

Report Name	Outline Project Specification
Author (User Id)	Katherine Rose Farmer (krf)
Supervisor	Bernie Tiddeman (bpt)
Module	CS39440
Degree Scheme	G451 (Computer Graphics, Vision and Games inc. Industrial Placement)
Date	February 4, 2015
Revision	1.0
Status	Draft

1. Project description

The aim of the project is to produce a working model of a spherical or cylindrical invisibility cloak using recent research and advances in the theory of invisibility. This model will be created in a 3D graphics language and will be used to obtain results about how well current theories are working towards creating an invisibility cloak against visible light, what drawbacks are there and what is causing them, and how does the cloak affect visibility from inside the cloak and therefore it's use for moving around invisible.

It is essential that the cloak is created in such a manner that it takes into account all the mathematical theories about how such a cloak is possible, looking mostly into transformation optics and the ideas presented there, and also the idea of metamaterials, as these are a key part in creating a real cloak of invisibility. Metamaterials are materials where the structure is designed to cause some kind of effect, such as the metamaterials designed to cause light to negatively refract or to be invisible from radio waves without disrupting the electromagnetic field.

The end goal is to see if we can model the invisibility cloak, how well the invisibility cloak works and what strengths and limitations exist within current theory. There is also room for creating multiple models if possible, looking at comparisons between spherical and cylindrical models, different global illumination strategies and how these affect the invisibility cloak's strengths and weaknesses.

2. Proposed tasks

The first task that needs completing is research. This research will aim to give me a better understanding of the current theories surrounding invisibility. This will include research into current prototypes from invisibility cloaks from visible and non-visible light, what metamaterials have been used in these prototypes and the mathematics behind the invisibility. This should provide a basis for the model.

There will also need to be research into what programming language to use and any libraries that will need to learn for that programming language. This will also affect my decision on which type of global illumination technique to use. As the idea of multiple models is still dependant on the amount of time it takes to create one model and test it adequately, choosing one global illumination strategy now will give me a solid goal for my first model. Both of these will require some time to research and become familiar with, depending on whether or not I have previous experience.

Next will need to be a design of the cloak and tests needed based on previous research. This will include the necessary algorithms and formulas for the model, tests for the model, as well the various features that will be implemented.

Then the model will be implemented based on the design created. Depending on the amount of time it takes to complete these steps, multiple models may be created for contrast and comparison. These will all be tested and the results noted down for the final report.

3. Project deliverables

- Model(s) of invisibility cloak which will be implemented in the programming language chosen.

- Results of experiments and tests on model(s) in table format and linking to tests specified in the test specification.
- Design specification created using research
- Test specification including tests for robustness as well as experiments checking for limitations and strengths of the cloak.
- Final report
- Blog for progress tracking that will include links to current versions of program and any research undertaken with references to research.

4. Initial annotated bibliography

[1] U. Leonhardt and T. G. Philbin, "Chapter 2 Transformation Optics and the Geometry of Light," *Prog. Opt.*, vol. 53, pp. 69–152, May 2009.

This paper explains extremely clearly the mathmatics and theory behind transformation optics, which is the basis for invisibility research.

[2] D. Schurig, J. J. Mock, B. J. Justice, S. a Cummer, J. B. Pendry, a F. Starr, and D. R. Smith, "Metamaterial electromagnetic cloak at microwave frequencies.," *Science*, vol. 314, no. 5801, pp. 977–980, 2006.

This is a paper discussing an electromagnetic invisiblity cloak that hides objects from microwaves. The paper has been useful in outlining the type of structure the model might take, based on the kind of structure used in the metamaterial.

[3] J. B. Pendry, *Metamaterials and the Science of Invisibility*. London, UK: Youtube, 2013.

A lecture given by a leading expert in the field of metamaterials and invisiblity research. It explains the history of the subject, current advances in the field and the general theories, as well as projects he has worked on.

Outline Project Specification – 0.2 (Draft) Katherine Rose Farmer (krf)

E. Design Specification

Rendering Invisibility

Report Name	Design Specification
Author (User Id)	Katherine Rose Farmer (krf)
Supervisor	Bernie Tiddeman (bpt)
Module	CS39440
Degree Scheme	G451 (Computer Graphics, Vision and Games inc. Industrial Placement)
Date	February 23, 2015
Revision	1.0
Status	Final

1. Introduction

The purpose of this document is to lay out a design for the system as a whole and for the invisibility cloak model(s). This specification is intended to inform me and my supervisor of the design I have envisioned for the project and as a reference for my future work. Another aim of this specification is to work out the feature list which I will be building from, according to the feature driven development methodology I have chosen to work through. There will also be a section working through considerations I have had to make through research and limitations of what I have to work with. There will also be two overall designs, detailing the system design, including what I have already implemented, and the model design, which will be a design of different models I will aim to work through, applying previous research to creating them.

2. Design Considerations

Assumptions and Dependencies

The language I am using for this project is Javascript, mainly working with WebGL and Three.js as this is a graphics project. Therefore I am making the assumption that it will work on all operating systems but may not work on all web browsers. I am expecting it to run on Chrome, Safari and Firefox due to these having good support for WebGL. I know that Opera has it disabled by default but should support WebGL, and Internet Explorer has still got some issues with WebGL. I will make sure to include support for the three browsers as required, but I will make less of an effort to make it work for Opera and IE.

I will need to decide on what rendering technique I am using, as the research I have done has concluded for me that I need to be able to do complex refractions and the basic refraction support in WebGL or Three.js has some trouble modelling complex refractions. This means I am looking into real-time rendering techniques, as these will make for a better demo. I have decided that my final rendering technique it likely to be either ray tracing or photon mapping. I will intend to make the decision soon, after some more research into the dependencies of the two techniques.

Goals and Constraints

The goal of the system is to create a model or models of a theoretically possible invisibility cloak. This requires a large amount of research into current theories and therefore constrains the amount of implementation time I will have. I have taken this into account in my features, in that there will be an amount of optional features to be done if time is available.

I am also constrained by what kind of rendering technique I use and how that affects the rendering of refraction. If I cannot control the refraction ratio specifically, and also create negative refraction, I may not be able to accurately depict the invisibility cloak and will have to implement around the theory to produce a desired result.

Development Methods

The development methodology I am planning to follow is Feature Driven Development. The reason I have chosen this is due to the feature list and the

staggered planning/implementation stage at the end. The feature list works well with the staggered nature of this project, in that I have separate research and implementation requirements for each step.

I had considered the use of eXtreme Programming in this project but XP's practices were mainly focused around a small team and the communication within that team. I also feel the lack of design would be detrimental to my project as I need to research and plan the models I am planning on implementing.

3. Feature List

As I am following the Feature Driven Development methodology, I will be working through a list of features. These are the features I have come up with according to my research and prototyping.

1. Preliminary Research
    a. Research into current Invisibility Theory including metamaterials, invisibility cloaks and transformation optics.
2. Preliminary Learning of New Technologies.
    a. This will include WebGL, Three.js, Ray Tracing, Photon Mapping and any tutorials associated with these.
3. Outline Project Specification
    a. This outlines the projects and any deliverables I require from the project.
4. Prototype of Sphere Shell Structure
    a. Implementation of a spherical transparent shell structure with differing refraction ratios (if possible).
5. Prototype of Ray Tracer Rendering/Photon Mapping Rendering
    a. Implementation of a ray tracing engine or photon mapping engine that will enable a more realistic light model.
6. Design Specification.
    a. Discussion of feature list, model design and system design.
7. First Model – Spherical Invisibility Cloak
    a. First fully implemented model created based on design specification and early prototyping.
8. Test Specification
    a. Test specification detailing evaluations and tests to be used on current and future models.
9. Second Model – Cylindrical Invisibility Cloak
    a. Second fully implemented model created based on design specification and first model.
10. Optional – Viewpoint from within cloaks.
    a. The ability to look from inside the cloak and what kind of images would be shown.
11. Optional – Use of Polarized Light
    a. Taking polarized light into account to create a more realistic model.
12. Optional – Further Models
    a. Further models of invisibility cloaks using different geometrical structures and coloured light.
13. Project Diary
    a. Will take the form of a blog and track the progress of the project.
14. Final Report

a. Report detailing project progression, completion and results.

4. System Design

The system design is relatively simple as I am mainly interested in the creation of the model(s)

File Hierarchy

There will be a html file for each version, be that model or prototype. These will be published to a github page so that I can both display by webpages and have a decent version control system. I will also have a main weblog that contains a project diary, tracking my progress as required by FDD. There will also be a folder for each model/prototype that contains the JavaScript files necessary for the project, as well as any assets needed by the project. There will also be a folder for library files, such as three.js, that are used by all/most of the models/prototypes.

User Interaction

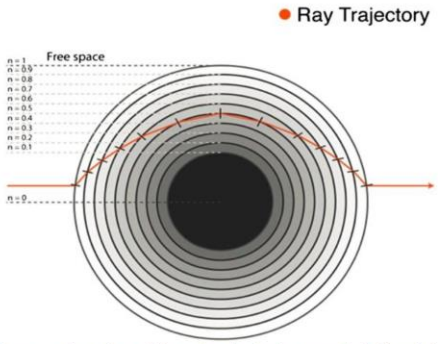
The user will be able to interact with the scene in a few ways. The user will be able to rotate the scene, as well as zoom in and out of the scene. The user will be able to change between invisible and non-invisible models, so that the difference can be seen. This will probably be done by using a shell model that contains normal refraction values and then switch these values to the invisibility values. The user should also be able to follow links through the system to access any version they wish to view.

5. Model Design

The design for the model(s) comes from the papers I have read and discussions with my supervisor. The model will be based on geometric structures and current theories on how an invisibility cloak would theoretically work.

Most models of the cloak show a layered structure that changes the refraction ratio through each layer, guiding the light waves around the inner object. This means that shell most also change from the back face to the front face of the object, having a negative refractive index at the back of the shell to guide the light back out of the shell at the correct point.

In a spherical model, the ray will need to curve around the center shell. In paper [2], values for a circular trajectory of the ray have been provided and it would be worth testing these values for a spherical model, though it is a cylindrical model that is being theorized.



Taken from [2] - These are the values of the refractive index at each shell and the trajectory the ray should follow.

For other geometrical shapes, they may be some more work involved, as the models may not have already been mapped by prior research. I have in one paper [1] equations for spherical, cylindrical, conical and ellipsoidal shells which should be enough geometrical shapes, though I would be intrigued to see how a cuboid shell would work, or if it even could work.

6. Bibliography

[1] M. M. Crosskey, A. T. Nixon, L. M. Schick, and G. Kovačič, "Invisibility cloaking via non-smooth transformation optics and ray tracing," *Phys. Lett. A*, vol. 375, no. 18, pp. 1903–1911, May 2011.

[2] K. S. Elassy, N. H. Rafat, M. E. Khedr, and M. H. Aly, "Fundamentals of designing cylindrical high-order transformation optics invisibility cloaks using silver–silica metamaterials," *Appl. Phys. A*, vol. 115, no. 2, pp. 531–539, Nov. 2013.

[3] D. Schurig, J. J. Mock, B. J. Justice, S. a Cummer, J. B. Pendry, a F. Starr, and D. R. Smith, "Metamaterial electromagnetic cloak at microwave frequencies.," *Science*, vol. 314, no. 5801, pp. 977–980, 2006.

F. Test Specification

Rendering Invisibility

Report Name	Test Specification
Author (User Id)	Katherine Rose Farmer (krf)
Supervisor	Bernie Tiddeman (bpt)
Module	CS39440
Degree Scheme	G451 (Computer Graphics, Vision and Games inc. Industrial Placement)
Date	April 6, 2015
Revision	1.0
Status	Final

Introduction

Goals and objectives

The goal of this document is to inform about the test process that will surround my system for modelling invisibility. The current state of the system will be discussed as well as the design discussed in the design specification. The tests will take these into account. The objectives of the test process are to create a robust system as well as validating the success of the research project against the theory discussed in the outline project specification and design specification.

Statement of scope

The scope of the testing will include the overall system, the models, the UI of the webpages. The scope of the validation will include the models only. Any behaviour that is specific to the theories and the models will not be tested, it will only be validated through experiments.

Major constraints

Due to the highly graphical nature of the project, there will be minimum unit testing and very little automated testing, if any at all.

Test Plan

Software to be tested

The software to be tested is identified by name. Exclusions are noted explicitly

Two websites will be tested. One can be found at the site: http://krf12.github.io which is the project diary. The other can be found at the site: http://krf12.github.io/RenderingInvisibility

Testing strategy

Unit testing

Components to be tested

Singular JavaScript files such as index.js

Tests required

JavaScript files needs to compile correctly and complete the functions required by the system. This will include drawing spheres correctly, loading shaders in correctly, shaders producing expected

effects etc. As there is some overlap between models and shaders, I am expecting to only test any unique functions and also test one file as a basis for the rest. If this file's functions work correctly, I will assume that any other models similar to it will work as well. An example would be testing only the shell sphere and expecting that – due to the shaders being identical to the shell cylinder example – that the other shell models won't need testing as well.

Expected results

Each file will produce the expected results, based on the validation testing as well as any effects and methods I have entered into the files that are unique to that file.

System testing

Components to be tested

The two complete websites - : http://krf12.github.io and http://krf12.github.io/RenderingInvisibility.

Tests required

The systems will require being navigated and ensuring that the system navigates correctly from page to page. There will also need to be checking on the project diary that ensures the dates are correct and there are no overlaps. The models on the project site will need to be checked for correct titles and navigation between models.

There will also be an expectation that the pages will load correctly. There may be some performance testing to record which pages are loading the slowest.

Expected results

That both systems will have correct navigation, perform within reasonable expectations, due to the process-heavy models being created.

Validation testing

Components to be tested

Each of the models will be tested individually.

Tests required

There is a need to compile a list of effects that the system is expected to be able to emulate and then test based on these effects. As of this version, the system is able to be tested on chromatic effects and perfect invisibility.

Expected results

The effects are created as expected, or that the results are understood as to the failures of the tests.

2.5 Test record keeping

The results for tests will be documented in the final report for this project and will likely be included as an appendix with important results highlighted in the relevant section.

The results of unit and system tests will likely be kept in table format. The results of the validation testing will be recorded the same way science experiments are recorded, using the test, expected results, actual results and conclusion structure.

2.6 Test metrics

There will be checking for the amount of tests that pass as expected and a limit of 75% pass rate for the project. If the first round of testing produces lower than this then another round of testing after defect fixing will be needed.

2.7 Testing tools and environment

The tests will be carried out on multiple browsers, with chrome and firefox being the main browsers due to their extensive support of WebGL. There will be minimal testing on tablet and smartphone environments as the hardware requirements for the models are possibly out of range for most smartphones and tablets.

There will likely only be manual testing but I will look into using an FPS meter to check the performance of each of the models.

G. Refraction.java Output

START OF NEXT METHOD
(0.5773502691896258, 0.5773502691896258, 0.5773502691896258)
(-223759.75796835296, -162570.8223394498, -380683.2428815497)
0.9/0.8
280232.2939104177
(-251729.6680622483, -182892.13179205806, -428268.5467554463)
(-0.47552798223715526, -0.3454909668280459, -0.8090173854434043)
(237.7641290737884, 172.74575140626317, 404.5084971874737)
1.670068964810183
0.3321409799411978
(158.63497591834516, 115.25505654522527, 269.8859407098943)
0.8/0.7
4.0732941940921696E16
(-287691.0492139981, -209019.57919092354, -489449.76772051014)
(-0.47552798223715526, -0.345490966828046, -0.8090173854434044)
(158.63497591834516, 115.25505654522527, 269.8859407098943)
1.8043935409093521
0.1977327240505942
(127.17362255702034, 92.39704538392684, 216.36068925894176)
0.7/0.6
3.563161833957192E16
(-335639.5574201212, -243856.1757254978, -571024.7290137098)
(-0.4755279822371553, -0.34549096682804603, -0.8090173854434043)
(127.17362255702034, 92.39704538392684, 216.36068925894176)
1.8606202387316682
0.14133169384820798
(109.13275189302217, 79.28958562969565, 185.66772676637237)
0.6/0.5
3.7784542482680736E16
(-402767.46890089306, -292627.41086823435, -685229.6748109184)
(-0.4755279822371553, -0.345490966828046, -0.8090173854434042)
(109.13275189302217, 79.28958562969565, 185.66772676637237)
1.8917347627363326
0.1100640267822617
(97.068823241274, 70.52462842636841, 165.14334540377794)
0.5/0.4
4.670716526780096E16
(-503459.33612611634, -365784.26358529297, -856537.093513648)
(-0.4755279822371553, -0.34549096682804603, -0.8090173854434042)
(97.068823241274, 70.52462842636841, 165.14334540377794)
1.9115183070919326
0.09015372252149993
(88.27483686752984, 64.13542334752862, 150.1821221370912)
0.4/0.3
6.867130504163924E16
(-671279.1148348219, -487712.3514470573, -1142049.4580181974)
(-0.47552798223715537, -0.34549096682804603, -0.8090173854434043)
(88.27483686752984, 64.13542334752862, 150.1821221370912)
1.9252125844810113

0.07635421957644806
(81.49837202230086, 59.21203345748009, 138.6533115158895)
0.3/0.2
1.31698806281988976E17
(-1006918.6722522327, -731568.5271705859, -1713074.1870272958)
(-0.4755279822371554, -0.3454909668280461, -0.8090173854434043)
(81.49837202230086, 59.21203345748009, 138.6533115158895)
1.935255007958828
0.06622325879599707
(76.06979322778315, 55.267940104280214, 129.41766161676406)
0.2/0.1
4.5895323362856154E17
(-2013837.3445044653, -1463137.0543411719, -3426148.3740545916)
(-0.4755279822371554, -0.3454909668280461, -0.8090173854434043)
(76.06979322778315, 55.267940104280214, 129.41766161676406)
1.9429344349850026
0.0584682478560388
(71.59432241505559, 52.01632023228993, 121.8035358713942)
BREAK
0.1/0.2
(-0.4755279822371554, -0.3454909668280461, -0.8090173854434043)
(71.59432241505559, 52.01632023228993, 121.8035358713942)
1.9489968178586332
0.05234047477789308
(67.82215222808956, 49.275678173897695, 115.38593662020163)
0.2/0.3
(-0.47552798223715537, -0.34549096682804603, -0.8090173854434043)
(67.82215222808956, 49.275678173897695, 115.38593662020163)
1.9539039578263082
0.04737609671381577
(64.58647472511586, 46.924820865634054, 109.8810732415728)
0.3/0.4
(-0.4755279822371554, -0.3454909668280461, -0.8090173854434043)
(64.58647472511586, 46.924820865634054, 109.8810732415728)
1.9579570842473986
0.04327235405478223
(61.77108870845008, 44.879323195739794, 105.09125244960462)
0.4/0.5
(-0.4755279822371553, -0.34549096682804603, -0.8090173854434042)
(61.77108870845008, 44.879323195739794, 105.09125244960462)
1.9613610786430224
0.03982319968285239
(59.29222926239487, 43.07832639072396, 100.87396485749555)
0.5/0.6
(-0.47552798223715537, -0.3454909668280461, -0.8090173854434043)
(59.29222926239487, 43.07832639072396, 100.87396485749555)
1.9642601487754674
0.03688352349599287
(57.08778378375969, 41.47670300293098, 97.12353819279548)
0.6/0.7

(-0.4755279822371553, -0.34549096682804603, -0.8090173854434043)
(57.08778378375969, 41.47670300293098, 97.12353819279548)
1.966758741961779
0.03434816887960594
(55.11058942995182, 40.04018722154206, 93.75973423498273)
0.7/0.8
(-0.4755279822371553, -0.34549096682804603, -0.8090173854434043)
(55.11058942995182, 40.04018722154206, 93.75973423498273)
1.9689343648426314
0.03213906374699307
(53.32410365899364, 38.742229335780486, 90.7203830981803)
0.8/0.9
(-0.4755279822371553, -0.34549096682804603, -0.8090173854434043)
(53.32410365899364, 38.742229335780486, 90.7203830981803)
1.9708457430551325
0.030197025886566926
(51.699514789635636, 37.561896433955724, 87.95646742239437)
(-111877.4206811714, -81283.54560058781, -190337.6430285667)

Annotated Bibliography

[1] J. S. Choi and J. C. Howell, "Paraxial ray optics cloaking.," *Opt. Express*, vol. 22, no. 24, pp. 29465–78, Dec. 2014.

This is a paper detailing a recent cloaking device that does not use metamaterials. A counter example to the type of strcutures that this project was dealing with.

[2] T. Ebbesen, S. Hell, J. B. Pendry, and A. Brown, "2014 Kavli Prize in Nanoscience: A Discussion with the Laureates," 2014. [Online]. Available: http://www.kavliprize.org/events-and-features/2014-kavli-prize-nanoscience-discussion-laureates. [Accessed: 26-Apr-2015].

A highly informative discussion between three leading scientists in the areas surrounding the science of invisibility. It includes a great deal of detail on the current problems the area currently faces.

[3] Wikipedia, "Snell's_law," 2015.

A Wikipedia article on Snell's Law, the mathematical law dealing with refraction and how to calculate the angle of refraction, or the refraction vector.

[4] T. Sousa and NVIDIA, "Generic Refraction Simulation," in *GPU Gems 2*, Second., M. Pharr and R. Fernando, Eds. Pearson Education Inc., 2005.

An article on how to implement refractionin the NVidia Shaders. The method here was used as a basis for the refract method in the OpenGL ES implementation, and therefore as a basis for this project's refraction method.

[5] U. Leonhardt and T. G. Philbin, "Chapter 2 Transformation Optics and the Geometry of Light," *Prog. Opt.*, vol. 53, pp. 69–152, May 2009.

A highly deatiled paper on the basics of transformation optics and the different areas that lead up to its discovery and the basis of its theory. Also include a great amount of detail on metamaterials and how they relate to metamaterials. There is a lot of complicated maths contained in this paper but the authors have a talent at explaining it well that it can be understood by someone with a decent amount of mathematical experience.

[6] F. E. Wagner, S. Haslbeck, L. Stievano, S. Calogero, Q. A. Pankhurst, and K. P. Martinek, "Before striking gold in gold-ruby glass.," *Nature*, vol. 407, no. 6805, pp. 691–692, 2000.

A reference from the previous paper detailing an old form of metamaterial called ruby glass.

[7] J. B. Pendry, *Metamaterials and the Science of Invisibility*. London, UK: Youtube, 2013.

A lecture given by the leading figure in the Science of Invisivility, Sir John Pendry. This was the first piece of research given to the author and holds a great deal of knowledge about refraction, transformation optics, metamaterials and the reality of invisibiltiy and negative refraction.

[8] J. B. Pendry, "Negative Refraction Makes a Perfect Lens," *Phys. Rev. Lett.*, vol. 85, no. 18, pp. 3966–3969, Oct. 2000.

A paper by Sir John Pendry on how Negative Refraction, something that can only be achieved using a metamaterial with a refractive index of -1, and how that can be translated into a perfect lens that has no loss of resolution.

[9] J. Wang, Y. X. H. Chen, and B. Zhang, "Ultraviolet dielectric hyperlens with layered graphene and boron nitride," p. 20, May 2012.

A paper detailing a proof of concept hyperlens that is another example of what metamaterials are already achieving.

[10] D. Schurig, J. J. Mock, B. J. Justice, S. a Cummer, J. B. Pendry, a F. Starr, and D. R. Smith, "Metamaterial electromagnetic cloak at microwave frequencies.," *Science*, vol. 314, no. 5801, pp. 977–980, 2006.

A paper detailing the first proof of concept invisibiltiy cloak that worked on one plane and was designed to cloak at microwave frequencies. It was built use dielectric copper metamaterials with a split-ring structure.

[11] K. S. Elassy, N. H. Rafat, M. E. Khedr, and M. H. Aly, "Fundamentals of designing cylindrical high-order transformation optics invisibility cloaks using silver–silica metamaterials," *Appl. Phys. A*, vol. 115, no. 2, pp. 531–539, Nov. 2013.

A highly used paper in this project, it describes a design for a cylidnrical cloak. Not only interested in describing the type of metamaterial needed, it also describes a layered refractive index structure which was the inspiration for the shell structure used in this project.

[12] M. M. Crosskey, A. T. Nixon, L. M. Schick, and G. Kovačič, "Invisibility cloaking via non-smooth transformation optics and ray tracing," *Phys. Lett. A*, vol. 375, no. 18, pp. 1903–1911, May 2011.

Another highly used paper in this project, it describes multiple cloak designs and uses ray-tracing to path the ray through the cloak. It is what the project's main testing method was based on and where a lot of the refraction forumlas for the cloaks were taken from. This was the project that had the closest aims to what this project was aiming for, but also was still completely different., especially in the end product.

[13] E. Wallace, "WebGL Water," 2011. [Online]. Available: http://madebyevan.com/webgl-water/. [Accessed: 28-Apr-2015].

An example of what can be accomplished with real-time ray-tracing. This was the example that encouraged the author to attempt to use ray-tracing in the project, though they were ultimately unsuccessful.

[14] E. Gamma, R. Helm, R. E. Johnson, and J. Vlissides, *Design patterns: elements of reusable object-oriented software*, vol. 206. 1995.

The Gang of Four design pattern book, a highly regarded book. This was used to see if there were aiming design patterns that could be used for this project. The only one chosen was the MVC pattern.

[15] C. W. Tom Preston-Werner PJ Hyett, "GitHub," *2008*, 2013. [Online]. Available: https://github.com.

Github, the version control used for this project. It also provided the GitHub Pages, where the project blog and project website were hosted.

[16] Atom.io, "Atom.io." 2015.

A highly customizable text editor that was used during this project to edit and create the JavaScript and HTML files involved.

[17] Oracle, "Netbeans." 2015.

An IDE that is the author's most used IDE. Was used to create and edit the Refraction.java program.

[18] M. Sopylo, "WebGL With Three.js: Basics - Tuts+ Code Tutorial," 2013. [Online]. Available: http://code.tutsplus.com/tutorials/webgl-with-threejs-basics--net-35688. [Accessed: 03-Feb-2015].

The three.js tutorial that the author followed with great success.

[19] Stack Exchange, "Stack Overflow," 2013. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Stack_Overflow&oldid=571742484. [Accessed: 02-May-2015].

Stack Overflow was used many times in the project for help with the nuances of the three.js library. It is a forum of questions and answers, specifcally for coding and software issues.

[20] romano1la, "JavaScript — Skybox and environment map in Three.js," 2014. [Online]. Available: http://blog.romanliutikov.com/post/58705840698/skybox-and-environment-map-in-three-js. [Accessed: 02-May-2015].

Blog post detailing how to use skyboxs and texture maps in three.js

[21] Creative Commons, "Creative Commons — Attribution 3.0 Unported — CC BY 3.0," 2012. [Online]. Available: http://creativecommons.org/licenses/by/3.0/. [Accessed: 02-May-2015].

License that the skybox texture is covered under.

[22] Humus, "Måskonåive," 2010. [Online]. Available: http://www.humus.name/index.php?page=Textures&ID=77. [Accessed: 02-May-2015].

The skybox texture.

[23] L. Stemkoski, "javascript - three.js - What's the best way to put multiple textures/images on a single Sphere? - Stack Overflow," 2013. [Online]. Available: http://stackoverflow.com/questions/18305318/three-js-whats-the-best-way-to-put-multiple-textures-images-on-a-single-spher. [Accessed: 03-May-2015].

One stack overflow answer that helped to steer the author in the correct direction to creating a hemisphere, or using two materials on the same sphere. This one suggested using texture normals and lead the author to use ShaderMaterials to cover the sphere in multiple textures.

[24] WestLangley, "Create a concave half sphere with three.js - Stack Overflow," 2012. [Online]. Available: http://stackoverflow.com/questions/12425014/create-a-concave-half-sphere-with-three-js. [Accessed: 03-May-2015].

One stack overflow answer that helped to steer the author in the correct direction to creating a hemisphere, or using two materials on the same sphere. This one suggested using the extra variables for SphereGeometry.

[25] G. Profenza, "javascript - Three.js - Custom Shapes? - Stack Overflow," 2011. [Online]. Available: http://stackoverflow.com/questions/8284233/three-js-custom-shapes. [Accessed: 03-May-2015].

One stack overflow answer that helped to steer the author in the correct direction to creating a hemisphere, or using two materials on the same sphere. This answer suggested using LatheGeometry.

[26] DataArtsTeam and Google, "dat-gui." 2011.

The dat-gui library, which is explained in Appendix A.

[27] mrdoob, "three.js - JavaScript 3D library," 2013. [Online]. Available: http://threejs.org/. [Accessed: 26-Apr-2015].

The three.js library, which is explained in Appendix A.

[28] Open Source Initiative, "The MIT License (MIT) | Open Source Initiative." [Online]. Available: http://opensource.org/licenses/MIT. [Accessed: 06-May-2015].

The license that most of the third-party libraires fall under.

[29] "Apache License, Version 2.0," 2004. [Online]. Available: http://www.apache.org/licenses/LICENSE-2.0.html\npapers2://publication/uuid/6D643C22-56CE-4DE7-94F8-8A28C51B75F8. [Accessed: 06-May-2015].

The license that dat-gui falls under.

[30] Khronos Group, "refract - OpenGL 4 Reference Pages," 2011. [Online]. Available: https://www.opengl.org/sdk/docs/man/html/refract.xhtml. [Accessed: 29-Apr-2015].

The refracted method that is used by OpenGL which was used as a basis for the refract method used in this project.

[31] J. B. Pendry, "The Science of Invisibility," *Procedia Computer Science*, vol. 7. pp. 20–21, 2011.

A paper by Sir John Pendry on The Science of Invisibility. It discusses in more detail about the various different fields that combine together to create the science of invisibility.

[32] M. Gharghi, C. Gladden, T. Zentgraf, Y. Liu, X. Yin, J. Valentine, and X. Zhang, "A carpet cloak for visible light," *Nano Lett.*, vol. 11, no. 7, pp. 2825–2828, 2011.

An example of an invisiblity cloak that uses metamaterials and works on near-visible light. It is a flat invisibility cloak designed to disguise bumps in a carper at certain angles.