University of Dublin

TRINITY COLLEGE

Real-Time Rendering of Realistic Digital Humans

Kris Vanhoutte

B.A.I. Engineering

Final Year Project May 2016

Supervisor: Dr Rachel McDonnell

School of Computer Science and Statistics

O'Reilly Institute, Trinity College, Dublin 2, Ireland

DECLARATION

I hereby declare that this project is entirely my own work and that it has not been submitted as an exercise for a degree at this or any other university.

_____ _____

Name Date

Acknowledgements

I would like to thank my supervisor Rachel McDonnell for providing me with the opportunity to work on this project.

I would also like to thank Emma Carrigan, whose prior work gave me a great starting point, and Elena Kokkinara for answering all of my annoyingly simple questions.

Thank you to all of those who took the time to proof-read this document and provide feedback; Muireann, Tom, Deirdre and Jess.

Finally, a special thanks to Sean Forsyth for taking the time to share his expertise in 3dsMax and rendering in general, his advice had a huge impact on the final results.

Abstract

Rendering Digital Humans in real-time while retaining the level of detail that is attained in offline rendering is a complicated challenge to solve.

This project utilises an existing open-source model, Digital Emily, and renders her in real-time within the framework of Unreal Engine 4. To achieve this, the project relies on techniques, such as subsurface scattering approximations, which simulate the reality of light interaction with the human body in a computationally inexpensive way.

Physically-based shaders are created for rendering the skin, eyes and hair. These shaders are then applied to the model, and the results are evaluated on-screen and using a virtual reality headset.

Contents

1 Introduction **7**

1.1 Introduction 7

1.2 Objectives 8

1.3 Motivation 9

1.4 Challenges 9

1.5 Outline 10

2 Background and Related Work **11**

2.1 Background 11

2.1.1 Offline vs. Real-Time Rendering 11

2.1.2 Physically-Based Shaders 13

2.1.3 Subsurface Scattering 14

2.1.4 Skin 14

2.1.5 Eyes 16

2.2 State of the Art 17

2.2.1 Skin 17

2.2.2 Eyes 18

2.3 Digital Emily 21

2.4 Software Applications 22

2.4.1 Modelling Software 22

2.4.2 Unreal Engine 4 22

3 Design and Implementation 25
3.1 Rendering in 3dsMax 25
3.2 Model Export Optimisation 25
3.2.1 Mesh Smoothing 26
3.2.2 UV-Remapping 27
3.3 Shaders 27
3.3.1 Basis for Work 29
3.3.2 Eye Shader 29
3.3.3 Skin Shader 32
3.3.4 Other Shaders 34
3.4 Scene Creation 36
3.4.1 Scene Design 36
3.4.2 Lighting 36
3.5 Virtual Reality 37
3.5.1 Optimisation 37

4 Results 38
4.1 Physically-Based Shaders 38
4.1.1 Eye Shaders 39
4.1.2 Skin Shaders 42
4.1.3 Other Shaders 44
4.2 Virtual Reality 45
4.2.1 Resolution 45
4.2.2 FPS Performance 46

5 Conclusions and Future Work 47
5.1 Conclusions 47
5.2 Future Work 48

Bibliography **49**

Appendix **53**

Appendix A – Rendered Images 53

A.1 – 3dsMax Renders 53

A.2 – Unreal Engine 4 Scene 56

Appendix B – Textures 57

B.1 – Emily Textures 57

B.2 – Created Textures 59

B.3 – Other Textures 60

Electronic Resources (CD)

Chapter 1

Introduction

This chapter will provide an introduction and high level description of the background to the project. It will also detail the objectives of the project, the motivation behind these objectives, and the challenges associated with the rendering of realistic digital humans. Finally, it will outline the structure of the remainder of the report.

1.1 Introduction

A great many improvements have been made in the field of computer graphics in recent decades. Such improvements can be seen across the board from increased processing power, to a clearer understanding of the properties of light and materials, to new approaches and concepts when it comes to rendering, such as the use of physically-based shaders. These improvements have completely changed the graphics landscape: special effects and CGI characters in movies and games have begun to look truly realistic; and once futuristic technologies, such as virtual reality headsets, are finally making their way into the mainstream market.

Despite the advancements that have been made in technology, there are still many challenges when it comes to rendering realistic humans, both offline and in real-time. A primary obstacle is that because humans are exposed to each other on such a large scale, every day of their lives, they have become adept at picking up even the smallest of details that are out of place. This has led to a phenomenon known as the “Uncanny Valley” [0], where rendering looks almost, but not quite perfect, leading to a sense of unease or revulsion in the viewer. For this reason, many animators decide to render characters who are intentionally inhuman looking, whether that be through making them look more cartoon-like or through anthropomorphism.

In addition to the challenges which occur when trying to achieve realistic results, other barriers must be overcome while trying to carry out this rendering in real-time. The physics behind light interaction with parts of the human body, such as the skin, is well understood. Unfortunately, it would be impossible to simulate this in real-time with the current processing power available, as the calculations are too complex. Therefore approximations to the reality must be used, which provide comparable results while being less computationally expensive.

With the dawn of the Virtual Reality Revolution [1], and current trends in video games striving for immersion through the use of realism, the demand is increasing for the ability to render realistic looking humans in real-time. Content creators are no longer satisfied with settling for the limitations of current methods which result in less realistic characters and environments just because that is what they have done in the past.

1.2 Objectives

The objective of this project is to create a real-time rendering of a digital human head through the use of physically-based shaders. To achieve this objective a number of distinct subtasks were identified:

1. Set-up a pre-existing model in a 3D modelling application.
2. Export that model with the optimal settings, and import it into a gaming engine.
3. Create shaders within the gaming engine for each of the distinct parts of the head; the eyes, the skin and the hair.
4. Create a scene for displaying the model on-screen and in a virtual reality environment.
5. Optimise the lighting and shaders for this environment.

Based on the objective and subtasks above, measures of success for the project were established.

1. Achieving real-time rendering of the model.
2. Producing shaders which provide favourable results over the use of a basic texture and lighting model.
3. Fine-tuning those shaders so that they offer comparable results to those of the offline render.

1.3 Motivation

Much research has been done into rendering realistic human skin, but less work has been published on the rendering of realistic human eyes. While some groups have previously implemented the Digital Emily model in gaming engine environments, including Epic Games, the creators of Unreal Engine, her eyes are often done very simply, or left out entirely, due to the way in which they were created. The challenge for this project was to implement some of the previous work which has been done with skin, and try to create a novel shader for the eyes that provides a satisfactory result. As the only hair mesh available with the model is for the eyelashes, less focus has been placed on that aspect in this project.

The complexity of rendering digital humans, due to the multifaceted nature of the materials and the need for very accurate portrayals, provided a fantastic opportunity to gain understanding of the rendering process as a whole, and particularly rendering within a gaming engine. The requirement to use multiple different software applications to reach the final result gave insight into the difficulties that can arise when interfacing between applications, and the diverse methods by which creators of these applications approach rendering. Finally, this project allowed for exploration of the emerging technology of virtual reality headsets.

1.4 Challenges

The field of computer graphics, specifically the branch of that field related to the rendering of realistic digital humans, presents many challenges. These range from artistic ones, like getting the colours of textures correct, to technical ones, such as getting light to reflect just the right amount from a person's skin. A number of the broad challenges that are faced are detailed below.

- Models; creating a realistic model poses a number of difficulties and provides an opportunity for a number of different design decisions to be made. For example, a model which accurately portrays the different layers of the human skin would not be practical to render. This forces the designer to make a choice as to how detailed they want to make the skin, and whether or not they want to use additional textures, such as normal maps, to compensate for lack of detail.
- Textures; getting textures right is yet another complex issue. The textures needed, or desired, for a particular render, will largely depend on the shaders implemented for that render. Ideally, a full complement of textures would be available for every model, but

unfortunately this is often not the case, and so accommodations must be made to account for this when creating shaders.

- Shaders; understanding how light interacts with objects is no simple matter. Many papers have been written on techniques that can be applied when creating physically-based shaders, and which of these provide the best results without having a detrimental effect on performance. There could also potentially be challenges when trying to implement shaders which require textures that are not available.

Some of the technical challenges which were encountered during this project are listed below:

- Learning 3dsMax and Unreal Engine 4.
- Interacting between the different software applications.
- Adjusting techniques to account for lack of textures.
- Optimising lighting and shaders for virtual reality.

1.5 Outline

Chapter 2 of this report provides a background on the techniques used to simulate how light interacts with the skin and eyes. In addition it will introduce some examples of the state of the art work within the field of rendering digital humans, and explain the choice of technologies used for this project.

Chapter 3 details the design choices and implementation of the project. It will explain the particulars of the project in the context of 3dsMax and Unreal Engine.

Chapter 4 describes the end results of the project, providing a comparison between the various approaches taken within the project. It will also provide a discussion of some of the negative aspects of the results and how they could be improved.

Chapter 5 completes the report with a set of conclusions which have been reached as a result of working on this project. Potential future work which could be conducted is also discussed.

Chapter 2

Background and Related Work

This chapter will detail the background to the project. It will explain the techniques used for the rendering of the skin and eyes, and provide examples of the cutting edge work being carried out. Furthermore it will discuss the choice of model, modelling software and gaming engine used within this project.

2.1 Background

This section will begin by comparing offline and real-time rendering within the frameworks of 3dsMax and Unreal Engine 4. It will then introduce two important topics, Physically-Based Shaders and Subsurface Scattering, which will be referenced throughout the rest of this document. Following this, an explanation will be provided regarding the structure of human skin and eyes, and the interaction of light with the materials they're made of.

2.1.1 Offline vs. Real-Time Rendering

Rendering can be split into two main categories: real-time rendering (also known as online rendering), and pre-rendering (also called offline rendering). Real-time rendering involves rendering a scene with a frame-rate high enough that viewers can interact with environment. As such, each frame typically needs to be rendered in a matter of milliseconds. Offline rendering is not bound to this same time requirement, and so a single frame can take hours or even days to complete. As time is not a limiting factor, offline renderers can use algorithms which are too computationally expensive for real-time renderers to use.

Rendering in 3dsMax is an example of an offline rendering process. As there is no major performance requirement, this rendering is software based and runs on the CPU,

allowing for much greater flexibility in terms of programming. 3dsMax also uses a number of computationally expensive techniques to increase the realism of its renders. The most relevant of these techniques to this project, in terms of contrasts between 3dsMax and UE4, is the use of ray tracing to calculate the effect of lighting.

Ray tracing is a technique in which the path of light is traced from the viewer through pixels in an image plane and the colour of the object it intersects with is accurately calculated. The basic algorithm behind ray tracing is as follows:

- The ray is traced from the viewer (camera) through a pixel of the image plane, and through the scene.
- The ray is tested for intersection with the objects in the scene, and the nearest object is identified.
- From this point, a reflected ray is traced if the surface is specular, a refracted ray is traced is the surface is transparent, and shadow rays are traced towards light sources to determine which sources are visible to the point being shaded.
- The refracted and reflected rays will themselves hit surfaces, and the illumination at these points will be recursively evaluated, using limitations to prevent infinite recursion.
- The contribution of each traced ray is accounted for in calculating the illumination of the original pixel.

Ray tracing is based on the realistic simulation of lighting, and so can more accurately simulate effects such as reflections and shadows which can be difficult for less powerful algorithms. Ray tracing also has the capability to accurately simulate other material properties such as subsurface scattering, which will be discussed in Section 2.1.3.

On the other hand, rendering in Unreal Engine 4 is done in real-time. To reach desired frame-rates of over 60 frames per second, each frame must be rendered in under 20 milliseconds. To achieve this, rendering is carried out on the GPU, which somewhat limits the flexibility for rendering by comparison with a CPU, but provides much faster graphics processing speeds. While ray tracing is an accurate calculation of the effects of light rays, real-time rendering relies on approximations, and calculations are done on fragments rather than individual pixels. The set of calculations applied to each fragment are defined by the shader.

UE4 attempts to make the process of creating shaders as simple as possible for the user, and so uses intuitive-sounding descriptions for shader parameters such as “BaseColour”. The functionality behind these parameters are then hidden from the user unless they wish to delve

into the source code. UE4's shaders are all physically-based (discussed in Section 2.1.2), but they use less computationally expensive approximations than ray tracing for the calculation of shading and lighting. Some of these approximations are detailed in a presentation given by Brian Karis in 2013 [2]. There have been a number of updates to UE4 since this presentation was given, but no more recent documentation has been published detailing the techniques which are behind their shading and lighting models.

It should be noted that not all lighting in UE4 is calculated in real-time. UE4 allows for the use of three different types of lights; static, stationary and movable. If using static lights and static models, as is the case in this project, global illumination effects such as translucent shadows (the shadows which occur when light passes through a semi translucent material) and colour bleeding (objects are coloured by the reflection of coloured light from nearby surfaces) are calculated using Lightmass [3]. Lightmass precomputes lightmaps which are then used when the scene is rendered.

UE4 is designed for use with a variety of different rendering pipelines including Microsoft's DirextX11, OpenGL, Vulkan and Javascript/WebGL. For this project DirectX 11 was used. Details of how the rendering pipeline is implemented in DirectX can be found on the Microsoft MSDN website [4].

2.1.2 Physically-Based Shaders

Physically-based shading simulates the interactions between materials and light in a way that approximates how light acts in reality, as opposed to how we might intuitively think it should act [5]. It is done to achieve a consistent, plausible look under different lighting conditions using the principles of physics.

Physically-based shading describes the basic aspects of surface-light interaction such as specular reflection, when light bounces off a surface in a single direction, and diffusion, the behaviour of the portion of light which penetrates the surface. This is then further enhanced by taking translucency and transparency into account.

Other principles which are applied include energy conservation, where objects never reflect more light than they receive, and Fresnel reflections, the principle that all surfaces become more reflective at grazing angles or that the 'edges' are more reflective.

One of the more significant benefits of physically-based shading is the fact that these principles can be applied at micro-surface level, meaning that every tiny imperfection of the surface can be taken into account when rendering [6].

2.1.3 Subsurface Scattering

The phenomenon of light entering a translucent material, scattering and exiting in a different place is known as subsurface scattering. Absorption and scattering can vary for different wavelengths of light thereby influencing the appearance of colour for a particular material. An example of such a material is human skin. Only 5-7% of incident light is reflected by the outermost layer of the skin [7], and so the colour which skin appears to be to a viewer is largely as a result of the subsurface scattered light which has made its way back to the surface.

There are two major components of subsurface scattering. The first is forward scattering, which occurs when light enters an object from the front and reflects back to the viewer. The second is called back scattering, which arises when light illuminates the backside of an object, and the light rays pass completely through the object [8].

One of the more obvious effects of forward scattering is a general blurring of the diffuse lighting, and common approaches to subsurface scattering estimate it by implementing blurring in texture space. Much research has gone into improving the results which are achieved when approximating subsurface scattering. Some of this research will be discussed in detail in Section 2.2.1.

2.1.4 Skin

Human skin (Figure 2.1) consists of a number of different, translucent layers. The epidermis, the outermost layer of the skin, creates skin tone. The dermis, sitting beneath the epidermis, contains connective tissue, hair follicles and sweat glands. The last layer is the hypodermis, which is made of fat and connective tissue [9].

When light hits the skin, a number of events occur. Part of the light reflects off the epidermis; some of this will reflect back towards the person viewing it, while some will reflect onto other parts of the face – i.e. global illumination. Part of the light is absorbed. The rest of the light will enter the skin and scatter, and the process will repeat itself at the dermis-hypodermis boundary. Some light will reflect back and exit the epidermis at a different point to its original entry – subsurface scattering – and the rest will pass through the boundary and scatter again. In actuality the interaction of light is more complicated than this, as there is scattering which occurs other than at the boundaries, as discussed by Krishnaswamy and Baranoski (2004) [7], but this is enough detail for the purpose of this paper.

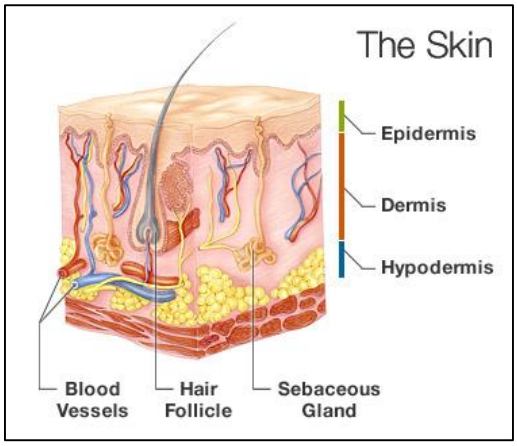


Figure 2.1 - Cross-section of human skin [10]

Unreal Engine 4 Subsurface Profile

Subsurface Profile is a shading model in Unreal Engine 4 designed specifically for skin and wax like surfaces. It is similar to the subsurface scattering shading model, primarily differing in the method in which it renders. The rendering for Subsurface Profile is based in Screen Space, which helps to better display the subtle subsurface effects seen in human skin, where backscattering is a secondary effect only seen in a few cases like the ears [11].

Subsurface Profile stores data about the distance the light in the Subsurface should scatter, the colour of the Subsurface and the falloff colour of the light once it has exited the object. By default it is set up to simulate Caucasian skin. The default profile was used in this project. The most notable effects of using Subsurface Profile are the softening of features and shadows.

Figure 2.2 - Subsurface Profile [11]

2.1.5 Eyes

The human eye is an extremely complex piece of anatomy. For the purpose of rendering, the set of elements that comprise the eye which need to be considered can be reduced, and the functions of those elements simplified. The most important parts of the eye in this context are the cornea, sclera, iris and pupil, and of slightly lesser importance the limbal ring.

The cornea is a transparent structure found in the very front of the eye that helps to focus incoming light. Sitting behind the cornea is a coloured, ring-shaped membrane called the iris, which has an adjustable circular opening called the pupil. The sclera is the outermost layer of three tissue layers which make up the eye, and is what gives most of the eyeball its white colour [12]. The limbal ring is the dark ring which appears around the edge of the iris, and connects the cornea and sclera.

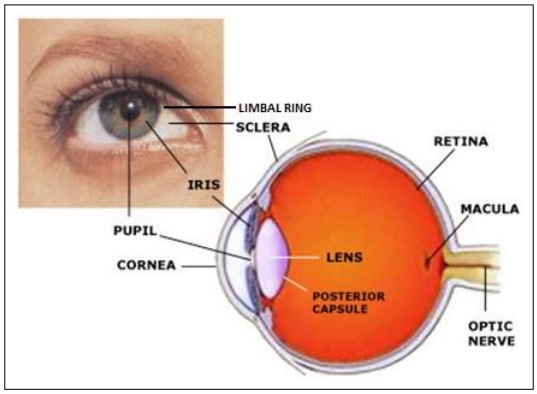


Figure 2.3 – Cross-section of a Human Eye [13]

Light Interaction

Each of the aforementioned parts of the eye interact with light in diverse ways. Both the cornea and sclera reflect a certain amount of incoming light. In the case of the sclera, these reflections are distorted, while reflections in the cornea are much sharper. The rest of the light which interacts with the sclera will appear as diffuse light, producing the colour of the sclera – primarily white, with thin red blood vessels running throughout, and additional redness at the edges of the eye.

The light which is not reflected back by the cornea will pass through it and experience refraction due to the large change in refractive index between air and the material of the cornea. This refracted light then continues on to the iris and pupil sitting behind it. The effect of the

refraction is relatively unnoticeable when looking at a front facing view of the eye, but hugely changes the appearance of the iris from a side angle.

Light which hits the iris will reflect back its diffuse colour. There is no specular reflection from the iris. Light which enters the pupil seems to "disappear", giving the pupil the appearance of a black hole.

The limbal ring appears as a dark ring around the iris when viewing the eye from the front, and vanishes when viewing the eye from the side. This effect occurs due to the refraction which is occurring in the cornea. This is an important element to take into consideration when attempting to model realistic human eyes, as it helps to reduce the often harsh boundary between sclera and iris.

Lastly, the eye as a whole has a wet look to it, due to the fluid which exists both within the layers of the eye, and on the surface of them. The wet look of the sclera and cornea can be accounted for with the specular reflections for these two elements. Additional wetness must be accounted for, in particular at the boundary between the eyeball and eyelids, when seeking realism with human eyes.

2.2 State of the Art

Much work has been completed regarding the real-time rendering of realistic humans, both open-source and proprietary. Enough so that an entire paper could be written solely on the results of this work. A small subsection of the work most relevant to this project has been selected for inclusion within this document. This subset is described in the following sections.

2.2.1 Skin

Vast amounts of research have explored optimally rendering human skin. A large portion of this research has been finding ways to improve the approximations of subsurface scattering in translucent materials. Accordingly, this section shall discuss some of the advances that have been made in this field, as well as an example of a state of the art model which has been created subsequently.

Subsurface Scattering

Craig Donner and Henrik Jensen (2005) [14] produced a multipole diffusion approximation, which expanded on previous work done by Jensen (2001, 2002) [15, 16] on a dipole diffusion approximation. This previous work provided a giant leap in making subsurface scattering

practical. Their new method still captured the subtle softness that translucency adds to skin's appearance, as well as additionally capturing the effects of discontinuities at the frontiers of multi-layered materials [17].

d'Eon and colleagues (2007) [18] observed that the reflectance and transmission profiles that Donner and Jensen's multipole model predicted could be approximated by a weighted sum of Gaussians. The results of their work visually match Donner and Jensen's offline renderings but are achieved at real-time frame rates [17].

Jorge Jimenez and Diego Guitierrez (2008) [19] further optimized the work done by d'Eon and colleagues. They carried out three optimizations; culling of the irradiance map, adjusting the irradiance map to be depth-based, and clipping of the irradiance map. Their results achieved speed-up factors between 1.10 and 2.77 when compared to the work done by d'Eon and colleagues.

The research mentioned in this project is just a subset of the work which has been done in advancing subsurface scattering simulation. This is still a very relevant field in computer graphics and advances continue to be made on a regular basis.

Digital Ira

Digital Ira is a project which was created as a collaboration between Activision and USC ICT. Their work was a furthering of the work which had been done on Digital Emily. They created "a real-time, photoreal digital human character which could be seen from any viewport, in any lighting" [20]. Some of the advanced techniques used for the real-time rendering of Digital Ira include "separable subsurface scattering [Jimenez et al. 2012] in screen-space, translucency, eye refraction and caustics, advanced shadow mapping and ambient occlusion, [and] a physically-based two-lobe specular reflection with microstructure" [20]. A demonstration of Digital Ira is available to download on the NVIDIA webpage [21].

2.2.2 Eyes

This section will discuss some of the pioneering work which has been accomplished in the field of eye rendering. Some of the most impressive work in this area has only been recently released, making it clear just how topical the work completed for this project is.

Next-Generation Character Rendering

Although not published as a paper, Jorge Jimenez presented a portion of his work with Activision Blizzard on the realistic rendering of eyes at GDC2013 [22]. An example of this work is shown below in Figure 2.4.

There are a number of notable features of this work. The reflections accurately simulate the reality of eye reflections, i.e. that reflections in the cornea are sharp, while those in the sclera are distorted sinusoidally, through the use of a purpose created normal map. The specular reflections from the layer of wetness surrounding the eyes hugely improves the realism of his results, softening the line between eyeball and skin. His calculations of the refraction in the cornea are also much more realistic than conventional parallax refraction techniques [22].

Figure 2.4 – Example of the results achieved by Jorge Jimenez [22]

Eye Shader in Unreal Engine 4.11

At the time of writing, documentation for the eye shader in Unreal Engine 4.11 (released April 2016) has yet to be published. According to Epic Games [23] the new shader is a physically-based shading model for the eyes which "approximates subsurface scattering through the sclera, caustics on the iris and specular on the wet layer". The shader is intended to be used in conjunction with their purpose built eye material and eyeball geometry. Together, the shader, material and geometry "additionally model the refraction though the cornea, darkening of the limbal ring, with controls for dilating the pupil".

Once documentation has been published on how this shader functions, it will potentially provide a basis for future open-source work on eye rendering, given the open-source nature of the engine and the community which revolves around it.

High-Quality Capture of Eyes

One of the common limitations when rendering realistic human eyes is that the models often fail to reflect the real geometry of a human eye. Instead simplified models are created which may be the incorrect shape – simple spheres in many cases – or have incorrect or unrealistic proportions. One such typical approach to creating an eye model is to start with a sphere and simply extrude a bulb on the front to represent the cornea. While current approaches are adequate in some cases and provide a generic eye model which can be reused, they are not ideal for a realistic render.

Bérard, Pascal, et al. published a paper in 2014 [12] which demonstrates the individuality of the human eye, and proposes "a novel capture system capable of accurately reconstructing all the visible parts of the eye; the white sclera, the transparent cornea and the non-rigidly deforming coloured iris". Their capture system involves the use of a six-camera system, which acquires approximately 140 images for a single eye dataset.

These images are used to generate high resolution texture maps for the sclera and iris. The texture map for the sclera contains all of the visible parts of the sclera as well as a synthetically created texture for the parts of the sclera which are not visible in any of the eye poses. The texture map for the iris is computed using numerous views to reduce the presence of artefacts, which would occur if a single view was used. The texture also takes into account multiple iris dilations to attenuate shading changes caused by the deformation of the iris.

In generating the texture map for the sclera, an initial model of the eye is also created, which begins as generic eyeball mesh generated by an artist, and deforms according to meshes generated from the sclera images. This eyeball mesh is then further supplemented with the addition of a cornea mesh, which is produced from a combination of reflection, refraction and position constraints. Lastly, an iris mesh is generated which accurately reproduces the topography of the iris under different instances of deformity.

2.3 Digital Emily

Digital Emily [24] is a freely available model created as part of the Wikihuman Project [25], a collaboration by the Digital Human League to advance the study of digital humans.

High resolution photography was used to capture detailed images of Emily O'Brien's head. These images were used to generate a 3D model, containing over seventy-five thousand vertices, as well as texture maps for the skin and eyes. The model consists of seven meshes, five for the eyes, one for the eyelashes, and one for the rest of the head. It's worth noting that the focus for the model was largely on the head mesh, and so the other meshes have much lower resolutions. This is also reflected in the texture maps, those for the eyes are simple PNG files, while those for the skin are EXR files, which contain much more data.

The five meshes which make up Emily's eyes are: two "inner eye" meshes, one for each eye, which appear to primarily be used for the iris and pupil (which is a physical hole in the centre of the mesh); two "outer eye" meshes, one for each eye, which represent the cornea and sclera; and a plica mesh, which corresponds to the layer between the eyeball and the eyelid, and also contains the tear duct. The plica is a single mesh, but has parts for both eyes.

The rendering of Emily was done in 3dsMax and Maya, and so depending on the graphics card in a system, it can take up to an hour and a half to render a single view, which is unworkable for real-time applications.

Figure 2.2 - Digital Emily [26, 27]

2.4 Software Applications

Two main software applications were needed for this project; a modelling software which could be used to open the model and export it in a more suitable format, and a gaming engine which would be used to render the model in real-time. The following sections explain the decisions which were made in the selection of these applications.

2.4.1 Modelling Software

As mentioned above, the data files for Digital Emily are available in both 3dsMax and Maya formats. 3dsMax and Maya are software applications provided by Autodesk used for modelling, animation and rendering. Historically, 3dsMax has been considered the superior modelling software, while Maya has been considered the better software for animation. Given that the main requirement for this project is to export the model for use in Unreal Engine 4, as well as creating an offline render, either of these applications would be more than adequate for the task. Both use the same rendering engine, and the main visible difference for this project would be the workspace layout.

Originally the Maya version of Digital Emily was selected, as it is the most recent version available on the website. However, the Maya version requires the use of an extra plug-in, a rendering toolkit called V-Ray 3.0, which does not come as part of the standard version of Maya. For this reason, in addition to a complication which occurred when attempting to import the model from Maya into Unreal Engine, it was decided to use the 3dsMax version instead.

2.4.2 Unreal Engine 4

Epic Games' Unreal Engine 4 (UE4) is a complete suite of game development tools. Due to its versatility and high quality rendering, it's not only used in the development of games, but also in digital films and 3D rendering.

While many game development studios use their own proprietary gaming engines, there's still a huge demand among indie developers and some larger studios for a purpose built gaming engine which has led to UE4 being used in some of the most popular games coming to the market; Tekken 7, Gears of War and Final Fantasy VII Remake, to name a few.

Why UE4?

UE4 was selected as the gaming engine to be used in this project for a number of reasons.

- It's available in full for free.
- While other gaming engines are also used in well-known games, UE4 is the most widely used non-proprietary gaming engine on the market right now.
- It already has built-in support for subsurface-scattering for skin, as well as documentation for setting up a skin shader. Documentation for hair and eye shaders has yet to be released at the time of writing, although both shaders have been included in the latest release of UE4 (4.11).
- As the creation of textures and meshes was not the objective of this project, good integration with existing rendering tools was essential. UE4 has well documented integration with both 3dsMax and Maya, as well as a number of free rendering tools such as Blender.

A number of other engines were considered for use:

- CryEngine was not available for free at the time this project was being implemented, although the latest version (Version V, released in March '16) has a "pay what you want" business model. At the time of evaluation CryEngine had a number of limitations when compared to UE4. The workflow appeared more cumbersome than UE4 and would have required a steeper learning curve. As well as this, textures need to be converted to TIF format before they can be imported, after which they're converted to DDS format, making it more difficult to test and tweak the assorted textures.
- Another option considered was Unity, which is similar to UE4 in terms of import pipeline. Apart from the difference in licensing costs both engines appear very similar. Unity's editor seems to be simpler to use, but UE4 has the edge when it comes to 3D graphics and physics capabilities, making it more suitable for this project.

UE4 Material Editor

Many material effects can be created in the main editor using the tools that are provided within Unreal Engine. Where further customisation is needed, the purpose built Material Editor can be used to create physically-based materials (shaders) which can then be applied to a geometry.

The Material Editor uses a node-based graphical interface. Each node is a material expression which can easily be configured and combined with other material expressions to create the desired effects. Once the material expression network is complete the material is

compiled. While this allows for quick creation and customisation of materials, the link between the material expression outputs and the final compiled material is somewhat of a “black-box”.

Chapter 3

Design and Implementation

This chapter details the implementation of the project, as well as the design decisions which were made to achieve the objectives as outlined in Section 1.5. The first two sections of this chapter discuss the elements of the project which utilised 3dsMax. The remaining three sections focus on the Unreal Engine related aspects of the implementation.

3.1 Rendering in 3dsMax

To begin, Digital Emily was rendered offline in 3dsMax. This was done for two reasons. Firstly, to confirm that she rendered correctly, and that all of the relevant textures and meshes were present in the downloaded data file before moving her into a real-time environment. Secondly, to have an "ideal" reference for how she looked from a number of different viewpoints, for comparison with the real-time results.

Once everything was configured correctly, a number of different renders were carried out, the results of which are shown in Appendix A. A view of the entire model was rendered using two different sets of background colours and lighting conditions. The purpose of this was to highlight the importance of environment on the quality of the final image. Zoomed-in renders of specific items such as the eyes, eyebrows, nose and ears, were captured from a number of different angles for ease of comparison between the 3dsMax and UE4 results.

3.2 Model Export Optimisation

Having validated the rendered results against the sample images online, the next step was to export the model from 3dsMax into Unreal Engine 4. There are a number of different ways this can be achieved. Meshes and textures can be exported individually or together, and can be

exported to FBX or OBJ formats. Following some experimenting it became clear that the shaders for Emily could not be converted into an importable format and shaders would have to be created from scratch in UE4 for each of the meshes.

FBX was selected as the format for export. Importing FBX files is straightforward and UE4 will automatically place the files in relevant folders. The following sections elaborate on some of the modifications which were made to the meshes before exporting from 3dsMax.

3.2.1 Mesh Smoothing

When Emily was first imported from 3dsMax into UE4, there was a noticeable tessellation problem with the eye meshes, both inner and outer. The problem was most noticeable for the iris, where there appeared to be very few polygons making up the iris, and the shading model which was being applied seemed to be flat shading [28]. This gave the iris a very facetted appearance, which was completely unrealistic.

To eliminate this effect, 3dsMax's TurboSmooth modifier was applied to all four eyeball meshes. TurboSmooth subdivides the geometry which it is applied to, while interpolating angles of new faces at corners and edges, and applies a single smoothing group to all faces in the object [29].

The improvement in the "outer eye" meshes was marginal by comparison with the "inner eye" meshes, as a large portion of the visible section of the "outer eye" is the transparent cornea. In contrast, the appearance of the iris improved dramatically, looking completely smooth after a single iteration of TurboSmooth.

Figure 3.1 – From left to right: tessellation, no tessellation, Original Shader (Section 3.3.1)

3.2.2 UV-Remapping

As will be further discussed in Section 3.3.2, a number of different shaders were created for the eye, one of which required the remapping of the texture coordinates (commonly referred to as UV-mapping) for the "inner eye" meshes. As neither the head mesh nor the eye meshes for Emily are axis-aligned within 3dsMax, there was no automatic way to perfectly re-align the UV-maps which meant that it had to be done manually.

The existing UV-map was "unwrapped" and aligned to the viewport, which had been adjusted to align the centre of the new map with the centre of the iris. This process executed separately for each eye, as they are separate meshes, and independently aligned. As a result of the manual alignment, there are imperfections in this new mapping, which is reflected in the end result, and discussed later in Section 4.1.1.

3.3 Shaders

The next task was to create shaders in Unreal Engine 4 for each of the meshes. The shaders in 3dsMax were used as a basis, alongside previous work done by Emma Carrigan and Jeremy Baldwin. Numerous approaches were taken for each shader. While some of them attempted to reproduce the methodology behind the 3dsMax shaders, others aimed to utilise the strengths of UE4 to create as realistic a result as possible.

Included on the CD attached to this document are the uasset files for all of the textures and materials used in the final scene. It is worth noting that when importing uasset files, the mappings between them may not carry through, and if so, will need to be remapped. This will specifically affect the inclusion of textures and material functions within materials.

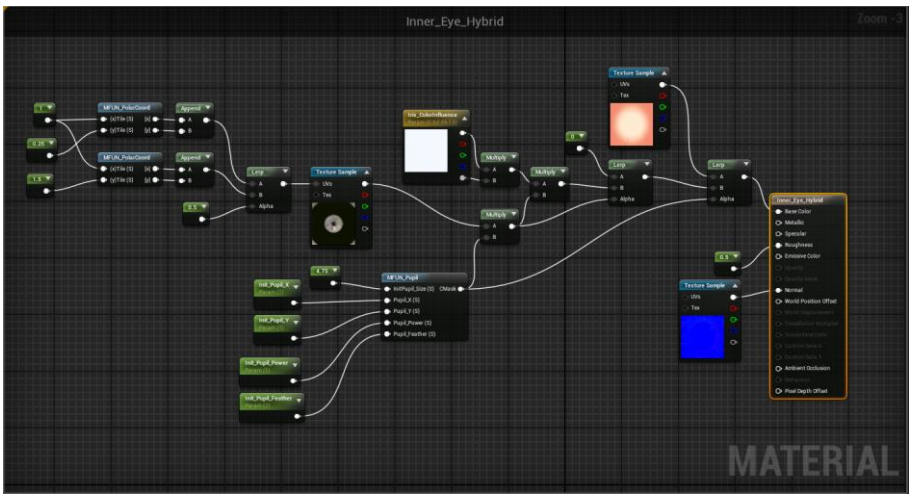


Figure 3.2 – Unreal Engine Material (Shader) Example

The material output parameters which are of relevance to this project are detailed in the following list:

- Base Colour; the base colour defines the overall colour (diffuse colour) of the material. It takes in an RBG colour vector as input. If a vector is input directly the colour of the material will be uniform. A texture or texture mask can be used to allow for different colours for distinct parts of the material. The base colour is also the colour which is used when calculating the colour bleeding effect of global illumination.
- Specular; this controls the amount of specular reflection which is visible on the object. It takes values in the range of 0 to 1, where 0 is no specular reflection, and 1 is maximum specular reflection. When using this parameter consideration much be given to the base colour as the specular reflections should have a colour which reflects this.
- Roughness; as the name suggests, this controls how rough the material is. A rough material will scatter reflected light in more directions than a smooth material. Like specular, it takes values in the range of 0 (completely smooth) to 1 (completely matte or diffuse).
- Opacity; this parameter is used when the material is translucent or a subsurface shading model is used. A value of 0 represents complete transparency, while 1 represents fully opaque.
- Normal; this parameter takes a normal map as input. It is used to add physical detail by changing the "normal" direction of each individual pixel. The normal can be used to provide a huge amount of extra detail which gives the illusion that the mesh has more polygons than are actually rendered. It is useful for providing detail for the minute imperfections of the skin.
- World Position Offset & World Displacement; both of these allow for vertices of the mesh to be manipulated in world space by the Material. These parameters affect the base and tessellation vertices respectively. The visual effect which can be seen in this project is the changing of the shape of the mesh.
- Subsurface Colour; this is only used when the subsurface shading model is enabled. It allows for a colour to be added to simulate the effects of absorption and scattering on the light which enters into a multi-layered translucent material such as skin.
- Refraction; this parameter takes in a texture or a value that simulates the index of refraction of the surface with respect to air. Increasing this value will increase the angle of refraction.

3.3.1 Basis for Work

While skin shaders are well documented, little documentation is available in UE4 on how to approach the rendering of eyes. Emma Carrigan, a PhD candidate in Trinity College, has carried out previous work implementing the Digital Emily model in UE4, using 3dsMax as a basis. Specifically, she created a shader for each of Emily's eye meshes which were used as reference for this project.

- The inner eye shader was an opaque material which used a subsurface scattering shading model.
- The outer eye shader was a blend of two material functions, with a texture mask to define which parts of the mesh these material functions were applied to.
- The plica shader was a transparent shader based off a glass material.

These shaders will henceforth be referred to as the "Inner Base Shader", "Outer Base Shader" and "Plica Base Shader" respectively. As one of the approaches to creating eye shaders in this project was to attempt to replicate the results of the 3dsMax shader, these three shaders were recreated and used as a starting point.

Further reference material was found on the website of Jeremy Baldwin, a 3D artist [30]. Although he does not use the same meshes as Emily, his tutorial demonstrates an approach which can be taken to render eyes in UE4. The shader which Jeremy has created is a fully contained eyeball shader, which has its own associated textures and material functions. A further feature of the shader is that it allows for the size of the iris to be dynamically adjusted. It was decided to use Jeremy's shader for reference due to the realistic eye colour, which at the time of discovery was far better than the results which were being achieved with Emily's own textures. This shader, henceforth called the Unreal Eye Shader, was taken as a starting point, and simplified for use in this project, as discussed below.

3.3.2 Eye Shader

Two distinct approaches were taken with the eye shaders. The first approach was to use Emily's original eye textures, and attempt to make them look as similar as possible to the rendering in 3dsMax, by using similar parameters to those in the 3dsMax shader where possible. The second approach was to use a combination of Emily's own textures and the simplified Unreal Eye Shader.

Original Shader

The first approach, referred to throughout the rest of this document as the "Original Shader", involved creating shaders for the inner eye and outer eye meshes. The details of these shaders, including further material functions used for the outer eye, are expanded on below.

Inner Eye Shader

As with the Inner Base Shader, the inner eye shader is an opaque material with a subsurface scattering shading model. Both the base colour and the subsurface colour attributes come from the inner eye colour texture (Appendix B.1), the former is scaled by a factor of 10, and the latter by a factor of 5. Scaling of 2:1 was used for the balance between these colours to achieve a blended result without losing detail. There is no correlation between these scaling factors and the parameters used in 3dsMax, due to the different functionality each application has for subsurface scattering.

All other parameters in this material come directly from the 3dsMax shader. The inner eye bump map (Appendix B.1) is used for the world position offset, scaled down by a factor of 0.1. Lastly, the specular reflection is set to 0. As discussed in Section 2.1.4, there is no specular reflection from the iris.

Outer Eye Shader

The Outer Eye Shader blends two material functions, a base material and a top material. This blending is accomplished using the standard material blending function within Unreal Engine. The blending function blends all attributes of the two input material functions.

A texture mask (Appendix B.2) was created for use as input to the alpha channel. The effect of using this texture mask is that the base material will be solely used for the bulbous part of the outer eye mesh which represents the cornea, and the top material will be used for the rest of the outer eye mesh which represents the sclera. A translucent material function is used for the base material, and a subsurface scattering material function is used for the top material. The mask which was used within 3dsMax was generated automatically from a gradient function, and so a new mask had to be manually created for this project.

Outer Eye Shader – Translucent Material Function

In the translucent material function, the specular attribute is set to 0.95 and the roughness to 0. Since the material is transparent, the choice of base colour is irrelevant, and so a pure black colour was used (RGBA 0, 0, 0, 0). These values all come directly from the 3dsMax shader.

The opacity is a linear interpolation between two values A and B (0.1 and 0.3) with an alpha input which is defined by a Fresnel function. Low values were selected for the opacity to reflect the transparency of the cornea, but values greater than zero were used so that it would still be visible as it is in reality. The alpha input determines the weighting given to each input value in the linear interpolation, according to the following function.

$$Output\ Value = (1 - alpha) \times A + (alpha) \times B$$

The Fresnel function calculates a fall-off based on the angle with which a piece of geometry is viewed; when the geometry is parallel to the camera, i.e. facing directly towards it, the Fresnel function has a value of 0, when the geometry is perpendicular to the camera, the Fresnel function has a value of 1.

The refraction attribute was set to a constant of 1.372, reflecting the true value for the index of refraction of the cornea [31].

Outer Eye Shader – Subsurface Material Function

As in 3dsMax, the subsurface material function makes use of the outer eye bump map (Appendix B.1), scaled by a factor of 0.038, used as input for the world displacement. A white colour vector (RGBA 255, 255, 255, 0) was used as the input for the specular attribute.

The base colour and subsurface colours both depend on the outer eye colour texture (Appendix B.1). The base colour uses this texture scaled by a factor of 4.5. The visual effect of this scaling is a reduction of the large amount of redness present in the texture, resulting in a more realistic sclera colour. For the subsurface input, the outer eye colour texture is used directly. The 3dsMax shader does not include a diffuse component (i.e. base colour), but its use is required in UE4 due to the differences in how subsurface scattering is implemented.

Hybrid Shader

The hybrid shader combines a variation of the eye shader Unreal Eye Shader with the outer eye shader discussed above. As mentioned in Section 3.2.2, the use of the Unreal Eye Shader required the remapping of texture coordinates for the inner eye, as the textures used in this shader require planar UV-mapping [32], in contrast to the spherical UV-mapping [33] used for Emily's own eye textures. The textures used in this shader are shown in Appendix B.3. The normal map is excluded as the details are not visible at such a scale.

Inner Eye Shader

The final shader used within this project is a simplified version of the Unreal Eye Shader. The dynamic resizing of the iris has been removed, along with a large number of extra nodes that were included within the original shader which are unnecessary for this static scene. The shader uses a normal map directly input without scaling as the normal input. The roughness is set to 0.5, which simulates the micro faceted nature of a realistic iris topography.

The only other attribute which is used is the base colour, which is where all of the complexity of the Unreal Eye Shader comes into play. Put simply, it contains two extra material functions. One of them defines the pupil size, and the other is used to generate polar coordinates, which transforms the spherical iris ring texture map into a planar version. A linear interpolation is carried out between the new iris texture map and a texture map for the sclera.

When applied to Emily's eye mesh, the sclera from Unreal Eye Shader did not provide as realistic a result as that achieved with Emily's own texture, and so it was decided to use a combination of the simplified Unreal Eye Shader for the inner eye, and the outer eye shader discussed in the previous section.

3.3.3 Skin Shader

Two different methods were used when creating skin shaders. The first emulated the shader in 3dsMax, creating one base material, and using material functions to imitate the method by which different shaders are nested within 3dsMax. The main reason for creating a 3dsMax-like shader was to see the results which would be achieved in UE4 when using a similar shader, but with UE4's renderer and lighting, as compared to the ray tracing which is present in 3dsMax. The second method used Unreal Engine documentation on setting up human skin [34] as a basis for work, and adjusted it to work with the textures which were available for Digital Emily.

3dsMax-like Shader

The material for the 3dsMax-like shader is an opaque material which uses the subsurface shading model. It consists of a blend between a diffuse material function as the base material, and a specular material function as the top material. A Fresnel function is used as the alpha channel input.

Diffuse Material Function

The diffuse material function is a further blend of a subsurface scattering material function as a base material, and a single scatter function as a top material. No alpha input is used for this blend. The single scatter material function is very basic, utilising only two of the material attributes. The roughness is set to zero. The base colour takes the single scatter texture (Appendix B.1) from Emily's texture set as its input. The texture is scaled by a factor of 0.82 before being input into the base colour attribute. This value comes directly from the 3dsMax shader.

The subsurface scattering function is only slightly more complex than the single scatter function. As a base colour it uses black (RGBA 0, 0, 0, 0). Its specular input is set to 0. The subsurface colour input comes from the diffuse texture (Appendix B.1). This texture is scaled by a factor of 1.2. Again, all of these values come directly from the 3dsMax shader.

An additional texture was included in this material function which was not present in the 3dsMax shader. A normal map (Appendix B.2) was created specifically for this project by Sean Forsyth using a 3D modelling software called ZBrush. This normal map was used as input to the normal output for the material function. The purpose of this was to make some of the fine imperfections of the skin visible.

Specular Material Function

The specular material function is also a blend of another two material functions, in this case two specular functions with different scales. Both specular functions have a base colour of black and a roughness of zero. The specular texture (Appendix B.1) was used as input for the specular output. For the base material this texture was scaled by a factor of 0.4, while for the top material it was scaled by 0.6. These scales come from the 3dsMax shader.

UE4 Skin Shader

As mentioned above, [34] provides information on how to set up human skin in Unreal Engine. Unfortunately, it assumes the presence of a number of textures which are not available for the Digital Emily model. As such, this documentation was used as a basis for creating a skin shader which better fit the existing textures.

The material is opaque, and uses the Subsurface Profile shading model discussed in Section 2.1.3. The base colour is a simplified version of that shown in [34]. The version in [34] uses a pore mask texture to scale the colour coming from the diffuse texture. As there was no pore mask texture for Emily, this scaling was left out. The base colour is hence made up of the diffuse texture (Appendix B.1) which is scaled by a power, and then multiplied by a vector. The default values which were present in [34] were used for the power (1) and vector (RGBA 1, 1, 1, 0). These currently do not change the original texture in any way, but they were left in to allow for potential future modifications.

The specular attribute in [34] depends on the pore mask texture which was mentioned above. As a work around, the specular texture from Emily (Appendix B.1) was linearly interpolated with a value of 1, using a Fresnel function as the alpha input. The result of this linear interpolation is then squared and used as the input for the specular attribute.

The normal map mentioned in the 3dsMax-like Shader was also used in this material. As there was no blending of attributes, the effects of the normal map were originally much more pronounced for this material. To counter this the normal map was "flattened" using the "FlattenNormal" node in UE4. The normal map was flattened by a factor of 0.55.

An opacity map was created for this project (Appendix B.2). It was created using Emily's specular texture map as a basis. The purpose of using an opacity mask was to simulate the transmission of light through thin sections of the skin, such as the nostrils and ear membrane. The opacity mask was used directly as an input for the opacity attribute of the material.

3.3.4 Other Shaders

There were two other meshes which needed to be accounted for; the eyelashes and the plica. Three approaches were taken to create a shader for the eyelashes, and a single shader was created for the plica mesh. These shaders are detailed below.

Eyelash Shader

Three different shaders were created for the eyelashes. The first was a simple shader which only took into account the diffuse element, and was set to the colour of Emily's eyelashes in 3dsMax which is pure black (RGBA 0, 0, 0, 0).

The second shader was slightly more complex. Inspired by a freely available hair material on unreal engine's forums [35], the aim of this shader was to include a specular lobe to replicate the shine that appears on real hair. Unfortunately UE4 does not provide inputs for two specular lobes, and the eyelashes make up such a small portion that they were deemed too insignificant to try and come up with a work around, so the specular reflection which is included is solely a white colour vector (RGBA 255, 255, 255, 0) scaled by a factor of 0.1. The purpose of this scaling was solely to reduce the amount of specular reflection to better simulate a realistic amount of specular reflection from hair.

The final shader attempted to make use of the translucency property for Unreal Engine materials. This was somewhat inspired by a discussion into the approach Epic Games were taking with their new hair shader in UE4.11. Furthermore, the 3dsMax shader makes use of transmission through the material to provide the secondary specular highlight which is present with hair. Unfortunately, there is currently no support for specular highlighting when using a translucent material, and so this shader did not provide the desired result.

The results of the first two shaders will be discussed in more detail in Chapter 4.

Plica Shader

As stated in Section 2.3, the plica mesh for Emily is a mesh which surrounds each eyeball, and sits between them and the eyelids. The plica shader that was used was largely the same as the Base Plica Shader, with only minor changes to a number of parameters to closer match those in 3dsMax. As the material is almost completely translucent, the base colour (diffuse colour) is irrelevant, and so was set to black. There are then only three parameters which have an effect on the final look of this material.

Opacity and refraction are both linear interpolations with alpha inputs defined by the same Fresnel function. The opacity is a linear interpolation between the values of 0.008 and 0.02, which result in a highly transparent material, and the refraction between the values of 1.05 and 0.95.

Finally, a bump map (shown in Appendix B.2) was used as an input to the world displacement output. The purpose of using this output was to reduce the smooth appearance of

the plica mesh and make it appear more realistic. The texture mask was scaled by a factor of 0.25, which comes from the 3dsMax shader.

3.4 Scene Creation

Once all of the shaders had been completed, a scene was created to display the model and allow the results to be evaluated both on-screen and using a virtual reality headset, the Oculus Rift. Images of the scene are show in Appendix A.2.

3.4.1 Scene Design

To maximise control over the environment an indoor scene was created. Consequently the light can be easily controlled, and the movement of the viewer can be easily directed. The scene which was created resembles a gallery space, with two plinths for displaying two different versions of the model. Online tutorials were used to aid in the creation of the scene [36], and the meshes and materials used within it come from the content examples package which is available for each version of Unreal Engine [37].

3.4.2 Lighting

Lighting is a key factor in real time rendering and to ensure maximum control over the effects of lighting, the decision was made to use ambient lighting. This is very similar to the way lighting is approached in film studios and theatre where natural light is either completely replaced or complemented (often by reflectors) to get the desired effect. The use of ambient light allows the use of a rendering technique call ambient occlusion. This technique calculates how exposed each point of an object is to the ambient light. By using an ambient occlusion map it is possible to control the atmosphere of the image. It creates non-directional shading and adds to the feeling of being indoors. In an outdoor environment ambient occlusion can be used to indicate an overcast day or a short time before sun set.

Based on lighting schemes used in theatre this project uses a simple three point lighting model. Three point lighting, as the name implies, involves the use of three different lights [38]. The “Key Light” creates the main illumination of the subject, defining the most visible lighting and shadows. In this scene, the Key Lights are situated to the left of, and slightly higher, than the model, when viewed from the front. The “Fill Light” softens and further extends the illumination provided by the Key Light, and increases the visibility of the subject. The Fill

Lights in the scene are situated to the right and at eye level to the model, when viewed from the front. Finally, the "Rim Light" – often called the "Back Light" – creates a bright line around the edge of the subject to help visually separate it from the background. The Rim Lights are situated directly behind, and above, the model.

All three lights used the same colour and similar intensities to give the effect of typical lights seen in a museum. To replicate the effect seen in film and theatre, the general rule is to make the Fill Light of a softer tone than the Key Light, while Rim Lights tend to be 'cooler' (more blue) which provides more depth to the scene.

3.5 Virtual Reality

The use of Virtual Reality (VR) headsets is easily integrated into Unreal Engine. The VR headset used in this project was the Oculus Rift DK2. The only prerequisite to begin using the Oculus with UE4 is to download the Oculus SDK from the developer website [39]. Once installed, any scene within UE4 can be run directly on the Oculus from Unreal Engine's editor.

3.5.1 Optimisation

A number of changes needed to be made to the scene in order for it to function in the desired manner within the Oculus. By default the Oculus places the camera at half its original height within the scene. To accommodate this, the height at which the camera is placed was elevated such that the viewer would be eye-level with the models of Emily in the scene. This is done within the Viewport of the "FirstPersonCharacter" blueprint [40] in the editor. The camera height is set to 210cm for use within the Oculus, and 75cm for use on-screen. The capsule half height is 145 for use within the Oculus, and 72.5 for use on-screen.

To avoid motion sickness, which is commonly experienced when using VR headsets, a number of changes were made to the motion of the character. Movement was limited to forward and backward movement. Strafing (moving left or right without turning), which is enabled by default, was removed. Rotation occurs via the viewer rotating their head, or in the case of on-screen, moving the mouse. Movement speed of the character was also reduced by scaling it by a factor of 0.24. This value was provided as an optimal value by Elena Kokkinara, from her previous experience in setting up scenes and characters for use in VR. All of these changes were implemented within the Event Graph of the character blueprint [40].

Chapter 4

Results

This chapter will detail the results which were achieved by this project. These results are evaluated in the context of the objectives outlined in Section 1.2. A discussion of the results is presented and possible improvements which could be made are proposed.

The physically-based shaders which were created during this project are compared with each other in the case where multiple shaders were created for a single mesh. The shaders are also compared with simple shaders, which only use the base textures supplied with Emily, and with the renders which were carried out offline in 3dsMax.

The final scene is assessed within the context of Virtual Reality. More specifically, this assessment considers the appearance of the scene given the resolution of the VR headset, and the performance of the scene in terms of frames per second (FPS).

4.1 Physically-Based Shaders

The main focus of this project was on the creation of physically-based shaders which work in real-time while providing accurate results. This section will evaluate the results of all of the shaders which were created as part of this project and discussed in Chapter 3; eye, skin, eyelash and plica respectively. The unsuccessful aspects of these results are also listed, and methods by which they could be improved are explored.

4.1.1 Eye Shaders

Original Shader

The Original Shader is shown in Figure 4.1 (c) below. As can be seen, the shader provides a relatively good result; the iris has the same detail as Emily's own eyes, and the sclera looks realistic.

There are two noticeable features which, if enhanced, would significantly improve this result. The first of these is the specular reflection of the cornea and sclera. At present there is no implementation of specular reflection for translucent materials within Unreal Engine, and so there was no feasible solution to this problem within the time-frame of the project. The new eye shader in UE4.11, which was discussed in Section 2.2.2 would most likely provide a much more realistic result for this aspect of the eyes, although some work may be required to adapt the shader to use the textures which are present for Emily's eyes.

The second feature which could be improved upon is the colour of Emily's iris. Replicating the colour of her iris as it appears in 3dsMax was not achieved in this project, and the resulting iris colour lacks realism. This issue was made even more prominent by the stark contrast between the iris and the sclera, which occurred due to the lack of a limbal ring.

Accomplishing a realistic colour for the iris should be a matter of optimising the scaling factors which were used for the inner eye shader of the Original Shader. Ideally a limbal ring would be a naturally occurring phenomenon due to the realistic simulation of refraction, but adding an artificial limbal ring to the iris or sclera textures could prove an adequate alternative.

3dsMax's ray tracing algorithm provides noticeably better results for both the colour of the iris and the reflection and refraction which occurs in the cornea. Ray tracing is able to accurately model the path of both the specular reflections on the cornea, and the light rays which pass through it. As mentioned above, UE4 cannot yet model the specular reflections for translucent materials, and the modelling of refraction doesn't yet provide accurate enough results for the limbal ring to become visible.

Hybrid Shader

The result of the Hybrid Shader, which can be seen in Figure 4.1 (e), is an eye which contains a realistic looking iris, both in terms of colour and texture detail, and a realistic looking sclera. One drawback of the Hybrid Shader is that it doesn't use Emily's own textures, and so therefore doesn't truly reflect the person on which she is based.

There are a number of noteworthy opportunities for improvement within the Hybrid Shader. The iris texture for the Hybrid Shader included a pupil, while Emily's eye models use a hole for the pupil. Due to the misalignment of the UV-remapping which was discussed Section 3.2.2, there is an obvious issue where both the "hole" pupil and the "texture" pupil are visible. This issue could be solved in a number of ways; the hole in the mesh could be filled, the UV-maps could be mapped more accurately (although this is not as simple as it might sound), or the iris texture could be scaled such that none of its pupil is visible outside the edge of the "hole" pupil.

Another opportunity for improvement is in the alignment of the cornea-sclera boundary and the boundary of the iris. As can be seen when comparing with Figure 4.1 (d), due to the relative sizes of the transparent cornea of the outer eye and the iris of the inner eye, a portion of the limbal ring is cut off. Understandably, this detracts from the realism of the final result. Similar to the pupil issue, this problem could be solved through correctly aligning the UV-maps and scaling the iris texture accordingly. It is worth noting that there's a difference in the colours of the sclerae of the inner and outer eye textures (those of the Unreal Eye Shader and Emily respectively), and so care should be taken that none of the sclera of the inner eye is visible to prevent a contrast from becoming apparent.

Comparison between Shaders

Both the Original Shader and the Hybrid Shader provide results which are much more realistic than the basic shader shown in Figure 4.1 (a) below. The most notable improvements they possess over the basic texture is the presence of detail in the textures for the iris and the sclera, which does not appear when these textures are directly applied to the base colour attribute in the basic shader.

The same can be said when comparing the Original Shader with the shaders created using the base shaders discussed in Section 3.3.1 (Figure 4.1 (b)). This result demonstrates clearly the progress that was made in this project, as the shader in Figure 4.1 (b) was used as a basis for the Original Shader.

Figure 4.1 (d) and Figure 4.1 (e) further illustrate the point which was made at the end of Section 3.3.1 regarding how applying solely the inner eye shader from the Hybrid Shader produces less realistic results than combining it with the outer eye shader from the Original Shader.

There is a clear contrast between the Original Shader and the Hybrid Shader below; specifically the iris and pupil details. Despite the issues with the Hybrid Shader, mentioned in the Individual Evaluation above, it provides the best result overall due to the presence of the limbal ring, as well as the more realistic colour of the iris.

Figure 4.1 - From top to bottom: (a) Shader using only the base textures, (b) Shader created using base shaders (Section 3.3.1), (c) Original Shader, (d) Hybrid Shader using transparent shader for the outer eye, (e) Hybrid Shader

4.1.2 Skin Shaders

3dsMax-like Shader

The 3dsMax-like Shader can be seen below in Figure 4.2 (b). Although this shader was created based on the shader implementation in 3dsMax, it doesn't quite match up to the standard within 3dsMax. The most noticeable differences are that in 3dsMax there are much more specular reflections visible. While it's clear that the ray tracing algorithm in 3dsMax provides greater accuracy for the calculation of specular reflections than the approximation used in UE4, a significant portion of the difference in specular reflections can be attributed to the different lighting conditions in the two scenes. The choices made with regards to lighting, i.e. position, number of lights and intensity, have a huge effect on the specular and diffuse lighting of the final render.

There is also a specular highlight present around the outline of the head, which should not be present. It's likely that both this highlight, and some of the lack of specular reflections throughout the rest of the face are due to the exponent which was used for the Fresnel function which determines the blend between diffuse and specular material functions, as described in Section 3.3.2. This result could be improved by optimising the Fresnel exponent.

UE4 Skin Shader

Figure 4.2 (c) shows the UE4 Skin Shader. The results achieved by this shader were of an extremely high quality. The specular reflections appear as expected on the skin, as does the specular lobe on the hair which appears due to the presence of the Rim Light. There is a softness to the skin both in terms of diffuse and specular lighting, which clearly demonstrates the effect of using the UE4 Subsurface Profile for implementing subsurface scattering. Minute details such as pores are visible thanks to the addition of the normal map, but require a closer viewpoint to be seen.

This shader provides results which are on-par with those of the 3dsMax render, and in some ways look even more realistic than the offline render. Specifically, the specular reflections of the UE4 Skin Shader look less wax-like than the 3dsMax render. This wax-like appearance of the specular reflections in 3dsMax is likely due to the intensity of the lights which are used, as opposed to any inaccuracy in the ray tracing algorithm.

The main improvement which could be made in this shader would be to find a way to account for back scattering in the thin skin surfaces such as the nostrils and ears. These are not visible in the 3dsMax render either, although it's unclear as to whether this is due to the lighting

setup in 3dsMax or whether backscattering has not been accounted for with the ray tracing algorithm which is being used.

Comparison between Shaders

The 3dsMax-like Shader and the UE4 Skin Shader both provide improved results over that of the basic shader shown in Figure 4.2 (a). Accounting for subsurface scattering significantly changes the colour with which specific parts of the face appear. The inclusion of specular reflections also significantly improves the result, which is most noticeable in the side profile.

Comparing the 3dsMax-like Shader and the UE4 Skin Shader, the perceptible differences are those of the colour of the skin and the specular reflections present. The UE4 Skin Shader gives better results in both of these cases. With regards to the colour of the skin, the 3dsMax-like Shader appears somewhat dull in comparison with that of the UE4 Skin Shader. The specular shading in the UE4 Skin Shader better reflects the amount of specular reflection which is visible with human skin.

Figure 4.2 – From left to right, two different views of: (a) Shader using only the base textures, (b) 3dsMax-like Shader, (c) UE4 Skin Shader

4.1.3 Other Shaders

The two other shaders which were created for the project were the Eyelash Shader and the Plica Shader. As there were no textures associated with the meshes these shaders are used for, they were only against the results from the 3dsMax render, and in the case of the Eyelash Shader, against other versions of itself.

Eyelash Shader

The second Eyelash Shader, as discussed in Section 3.3.3, included a specular lobe. This shader provided marginally better results than the first Eyelash Shader, which solely had a diffuse black colour, although the difference between the two is only noticeable in extreme close-ups. From a distance, neither shader fares up well when compared to the 3dsMax shaders, due to the presence of aliasing artefacts. An attempt was made to replicate the shader used in 3dsMax, but 3dsMax provides much greater anti-aliasing than is currently available in Unreal Engine.

Considering the relatively small proportion of the overall model that the eyelashes make up, much less focus was given to their associated shaders than to those of the eyes and skin. It is possible that with more work, a shader could be created which more closely mirrors that used within 3dsMax. The new hair shader included in the latest release of Unreal Engine (4.11) is also very likely to provide better results for eyelashes, as "it models two specular lobes, transmission and scattering" [23]. Unfortunately, as previously mentioned, documentation for this shader has yet to be released, and so it could not be included within this project.

Plica Shader

Similar to the Eyelash Shader, the Plica Shader was not given as much attention as the other shaders in this project. Unlike the Eyelash Shader, this was not solely due to the size of the associated mesh, but rather due to the current limitations of UE4. The Plica Shader needs to be translucent, but one of the key features, and the main contribution it has to increasing the realism of rendered eyes, is the specular highlights which should be present.

Unreal Engine's translucent materials don't currently allow for specular highlights to be included and so rather than spend time attempting to create a work around, this shader was made very similar to the Base Plica Shader, with a number of the parameters optimised.

The results for this shader are relatively imperceptible within 3dsMax. Upon close inspection it becomes apparent that this mesh has a number of faults in it, both in terms of its

transparency and its positioning. It could potentially be left out entirely from both 3dsMax and UE4 without having a huge impact on the model.

It would be worth investigating whether there has been any inclusion of a similar mesh with the new eye shader in UE4.11, and how it was handled if it is present, although this was not explored within the scope of this project. It should also be noted that the plica mesh which is included with Emily is not ideal, and could be improved upon through the use of blendshapes as discussed by Jorge Jimenez in his presentation which was discussed in Section 2.2.2 [22].

4.2 Virtual Reality

When viewing the result in the Oculus Rift it became very apparent that realism was not yet achieved. This section will discuss two of the limitations which were encountered with the Oculus, resolution and performance, which if addressed would greatly improve the final result.

4.2.1 Resolution

There is typically a trade-off between realism and performance with virtual reality; many of the amazing techniques which have been developed for rendering simply cannot be used because they will render too slowly. Surprisingly, with this project, the limiting factor was actually the resolution of the headset and not the frame-rate.

Testing for this project was done using the Oculus Rift DK2, which has a resolution of 960×1080 for each eye. Unfortunately, this resolution was too low for many of the minute details of the shaders to be discernible. In particular, the differences between two skin shaders discussed in Section 3.3.2 were almost unnoticeable. For this reason, only one of these shaders was displayed in the final scene (the UE4 Skin Shader). The differences between the eye shaders are more noticeable due to the stark difference between the two inner eye shaders, although the fine details of the sclera are also difficult for the viewer to detect.

Regrettably, there was not much that could be done to reduce the detrimental effect the resolution of the headset had on the result. A potential solution would have been to increase the size of the model, so finer details were more visible, but this would have introduced its own problems. One such problem is that increasing the size of the model would have reduced the realism of the experience for the viewer, as the size of the digital human head would have been totally disproportionate to the viewer's expectation of how large a human head should be.

It should be noted that the Oculus Rift DK2 is a developer version of the Oculus Rift. The retail version of the Oculus Rift is available for order now, with shipping expected to begin

in August 2016, while the HTC Vive is already on the market. Both the Oculus Rift Retail and HTC Vive have improved resolution over the DK2 of 2160 × 1200, or 1080 × 1200 for each eye. This increased resolution would provide better results for the scene created in this project. It is also likely that these resolutions will only increase further as the use of VR headsets becomes the norm, and so the results will continue to improve until frame-rate becomes the limiting factor.

4.2.2 FPS Performance

When run on-screen, both from within the UE4 editor, and having packaged the game for deployment, the scene achieved a consistent frame-rate of 62 frames per second. When run on the Oculus Rift, the average frame-rate was 50 frames per second, dropping to an average of 45 frames per second when the viewer moved very close to the head models.

A frame-rate drop when moving from on-screen to virtual reality was to be expected, as the scene has to be rendered twice for the stereo-vision of the Oculus Rift. This frame-rate is much lower than the target frame rate of the Oculus Rift DK2, which is 75 FPS.

There were no noticeable frame-rate issues when moving through or viewing the scene, on-screen or in the Oculus, although this can be attributed to the fact that it was a static scene with no dynamic objects other than the player. As a result of this, improving the frame-rate was not a focus for the scope of this project.

Improvement of this framerate would be a necessity if this project were to be expanded to include animation of the digital human model. Some ways in which the frame-rate could be improved are:

- Removing or reducing any lighting effects which are not going to be used in the scene.
- Increasing the target frame-rate setting within UE4.
- Reducing the quality of detail for the video settings such as view distance, anti-aliasing, textures, etc.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

The overall objective of this project was to create a real-time rendering of a digital human head through the use of physically-based shaders. This was achieved as outlined throughout this report.

Three measures of success were established in Section 1.2. The first two of these measures – achieving real-time rendering of the model and producing shaders which provide favourable results over the use of a basic texture and lighting model – were clearly accomplished as described in Chapter 4. The final measure – enhancing the shaders so they offer comparable results to those of the offline render – was accomplished only with the skin shaders and not the eye shaders. Despite this shortcoming, the project was considered a success overall.

Digital Emily originally seemed to be an excellent starting point and certainly provided a great deal of information. However, it became clear during the project that the primary focus for the model had prioritised the implementation of the skin and that weaknesses existed regarding other elements such as the eyes. These weaknesses were reflected in the final result.

If this project were to be repeated, it is recommended that less time be spent on trying to replicate the results of 3dsMax exactly, and more time be spent trying to optimise shaders using the potential of UE4.

There's a very steep learning curve involved with the majority of the tools used within this project. Although a huge amount of knowledge was learnt and understood over the course of the project, only the surface has been scratched when exploring the capabilities that these tools have to offer.

In conclusion, there's significantly more to creating realistic graphics than just having a nice shader. Ensuring the textures are correct, possessing the right textures, and having the ability to create new ones as needed is an enormous help. Understanding the meshes that all of these shaders are being applied to, and having the know-how to change those meshes would allow for much greater flexibility. Ideally, when working on a project of this nature, a team of artists, modelling software experts and programmers should be available to get everything functioning perfectly.

5.2 Future Work

Although the objectives of this project were achieved, there is much room for improvement throughout, some of which is discussed in Chapter 4. It was noted in Chapter 1 of this document that focus should be placed on creating a realistic eye shader, and this is still the case. Creating an optimal eye shader is one of the key requirements in rendering a realistic digital human. The work accomplished within this project in this area could be further improved upon by utilising the latest technologies which are currently being released for the task, such as the new eye shader in UE4.11.

Methods for skin shading have already reached such a high standard that rather than trying to improve the existing shader, time would likely be better spent elsewhere. As such, to improve the results for the skin within this project, a possible approach would be to attempt to replicate as accurately as possible the material network shown in [34]. This would require the creation of new textures to make up for those that Emily is missing, although the ones which are provided for the UE4 example could also potentially be used.

A major of focus of this project was into the creation of physically-based shaders. While having a realistic shader is important when attempting to render realistic digital humans, the effects of lighting should not be underestimated. There is a much room for improvement with regards to the lighting parameters in this project, from the exact placements of the lights, to the settings which control lightmap creation.

Bibliography

[0] Mori, Masahiro, Karl F. MacDorman, and Norri Kageki. "The uncanny valley [from the field]." Robotics & Automation Magazine, IEEE 19.2 (2012): 98-100. DOI: 10.1109/MRA.2012.2192811

[1] Laurence Cruz, The Network, (April 2016). *The Virtual Reality Revolution.* Available at: https://newsroom.cisco.com/feature-content?type=webcontent&articleId=1757975 (Accessed 19 May 2016) **News Article**

[2] Brian Karis, Epic Games, (2013). *Real Shading in Unreal Engine 4.* Available at: https://de45xmedrsdbp.cloudfront.net/Resources/files/2013SiggraphPresentationsNotes-26915738.pdf (Accessed 24 May 2016) **Presentation**

[3] Epic Games. *Lightmass Global Illumination.* Available at: https://docs.unrealengine.com/latest/INT/Engine/Rendering/LightingAndShadows/Lightmass/index.html#featuresforstaticandstationarylights (Accessed 24 May 2016) **Documentation**

[4] Microsoft. *Understand the Direct3D 11 rendering pipeline.* Available at: https://msdn.microsoft.com/en-us/library/windows/desktop/dn643746(v=vs.85).aspx (Accessed 24 May 2016) **Documentation**

[5] Epic Games. *Physically Based Materials.* Available at: https://docs.unrealengine.com/latest/INT/Engine/Rendering/Materials/PhysicallyBased/index.html (Accessed 15 May 2016) **Documentation**

[6] Jeff Russell, Marmoset. *Basic Theory of Physically-Based Rendering.* Available at: https://www.marmoset.co/toolbag/learn/pbr-theory (Accessed 17 May 2016) **Tutorial**

[7] Krishnaswamy, A. and Baranoski, G. V.G. (2004), A Biophysically-Based Spectral Model of Light Interaction with Human Skin. Computer Graphics Forum, 23: 331–340. doi:10.1111/j.1467-8659.2004.00764.x

[8] Digital Tutors, (2014). *Understanding Subsurface Scattering – Capturing the Appearance of Translucent Materials.* Available at: http://blog.digitaltutors.com/understanding-subsurface-scattering-capturing-appearance-translucent-materials/ (Accessed 19 May 2016) **Tutorial**

[9] WebMD, (2014). *The Skin (Human Anatomy): Picture, Definition, Function, and Skin Conditions.* Available at: http://www.webmd.com/skin-problems-and-treatments/picture-of-the-skin (Accessed 15 May 2016) **Article**

[10] WebMD. *Skin Information: Layers of Skin, Keeping Skin Healthy and More.* Available at: http://www.webmd.com/beauty/wrinkles/cosmetic-procedures-overview-skin (Accessed 14 May 2016) **Article (Image)**

[11] Epic Games. *Subsurface Profile Shading Model.* Available at: https://docs.unrealengine.com/latest/INT/Engine/Rendering/Materials/LightingModels/SubSurfaceProfile/ (Accessed 16 May 2016) **Documentation**

[12] Bérard, Pascal, et al. "High-quality capture of eyes." ACM Trans. Graph. 33.6 (2014): 223-1.

[13] SchoolWorkHelper. *The Human Eye: Parts, Diseases, Accommodation.* Available at: http://schoolworkhelper.net/the-human-eye-parts-diseases-accommodation (Accessed 4 April 2016) **Tutorial (Image)**

[14] Donner, Craig, and Henrik Wann Jensen. "Light diffusion in multi-layered translucent materials." ACM Transactions on Graphics (TOG) 24.3 (2005): 1032-1039.

[15] Jensen, Henrik Wann, et al. "A practical model for subsurface light transport." Proceedings of the 28th annual conference on Computer graphics and interactive techniques. ACM, 2001.

[16] Jensen, Henrik Wann, and Juan Buhler. "A rapid hierarchical rendering technique for translucent materials." ACM Transactions on Graphics (TOG) 21.3 (2002): 576-581.

[17] Jimenez, Jorge, et al. "Real time realistic skin translucency." IEEE computer graphics and applications 30.4 (2010): 32-41.

[18] d'Eon, Eugene, David Luebke, and Eric Enderton. "Efficient rendering of human skin." Proceedings of the 18th Eurographics conference on Rendering Techniques. Eurographics Association, 2007.

[19] Jimenez, Jorge and Gutierrez, Diego. “Faster Rendering of Human Skin,” Proc. Congreso Español de Informática Gráfica 2008 (CEIG 08), Eurographics Assoc., 2008, pp. 21–28.

[20] Alexander, Oleg, et al. "Digital ira: Creating a real-time photoreal digital actor." ACM SIGGRAPH 2013 Posters. ACM, 2013.

[21] NVIDIA, (2013). *Lifelike Human Face Rendering Demo.* Available at: http://www.nvidia.com/coolstuff/demos#!/lifelike-human-face-rendering (Accessed 20 May 2016) **Download**

[22] Jorge Jimenez, (2013). *Next Generation Character Rendering.* Available at: http://www.iryoku.com/publications (Accessed 24 March 2016) **PowerPoint**

[23] Alexander Paschall, Epic Games, (March 2016). *Unreal Engine 4.11 Released!* Available at: https://www.unrealengine.com/blog/unreal-engine-4-11-released (Accessed 2 April 2016) **Release Notes**

[24] USC ICT, (2015). *The Wikihuman Project.* Available at: http://gl.ict.usc.edu/Research/DigitalEmily2/ (Accessed 17 May 2016) **URL**

[25] Chaos Group. *Wikihuman.* Available at: http://www.wikihuman.org/ (Accessed 15 May 2016) **URL**

[26] USC ICT, (2015). *The Wikihuman Project.* Available at: http://gl.ict.usc.edu/Research/DigitalEmily2/images/Cam1_FlashReference_Eyes_v06.jpg (Accessed 15 May 2016) **Image**

[27] USC ICT, (2015). *The Wikihuman Project.* Available at: http://gl.ict.usc.edu/Research/DigitalEmily2/images/Cam3_FlashReference_Eyes_v06.jpg (Accessed 15 May 2016) **Image**

[28] *Shading.* Available at: http://ruh.li/GraphicsShading.html (Accessed 17 May 2016) **Tutorial**

[29] Autodesk, (2016). *TurboSmooth Modifier.* Available at: https://knowledge.autodesk.com/support/3ds-max/learn-explore/caas/CloudHelp/cloudhelp/2016/ENU/3DSMax/files/GUID-EA8FF838-B197-4565-9A85-71CE93DA4F68-htm.html (Accessed 17 May 2016) **Documentation**

[30] Jeremy Baldwin. *Unreal Eye.* Available at: http://www.jeremybaldwin3d.com/#!free-market-station/zwkhl (Accessed 3 April 2016) **Download**

[31] HyperPhysics. *The Cornea.* Available at: http://hyperphysics.phy-astr.gsu.edu/hbase/vision/eyescal.html (Accessed 19 May 2016) **Tutorial**

[32] Autodesk, (2016). *Planar UV Mapping.* Available at: https://knowledge.autodesk.com/support/maya-lt/learn-explore/caas/CloudHelp/cloudhelp/2016/ENU/MayaLT/files/GUID-B6519472-C0ED-4C07-99C6-12107A3509D9-htm.html (Accessed 18 May 2016) **Documentation**

[33] Autodesk, (2016). *Spherical UV Mapping.* Available at: https://knowledge.autodesk.com/support/maya-lt/learn-explore/caas/CloudHelp/cloudhelp/2016/ENU/MayaLT/files/GUID-2B74F60D-B6B5-4794-BDC1-17E2FB06F393-htm.html (Accessed 18 May 2016) **Documentation**

[34] Epic Games. *Material Editor – How To set up Human Skin.* Available at: https://docs.unrealengine.com/latest/INT/Engine/Rendering/Materials/HowTo/Human_Skin/index.html (Accessed 4 April 2016) **Documentation**

[35] MizuGames, Unreal Engine Forums, (2015). *Hair Material.* Available at: https://forums.unrealengine.com/showthread.php?63555-FREE-Hair-Material (Accessed 17 May 2016) **Forum**

[36] Unreal Engine, YouTube, (2015). *Introduction to UE4 Level Creation (v4.7).* Available at: https://www.youtube.com/playlist?list=PLZlv_N0_O1gak1_FoAJVrEGiLIploeF3F (Accessed 18 May 2016) **Tutorial Playlist**

[37] Epic Games. *Content Examples.* Available at: https://docs.unrealengine.com/latest/INT/Resources/ContentExamples/ (Accessed 19 May 2016) **Documentation**

[38] Jeremy Birn, 3dRender.com. *Three-Point Lighting for 3D Renderings.* Available at: http://www.3drender.com/light/3point.html (Accessed 18 May 2016) **Tutorial**

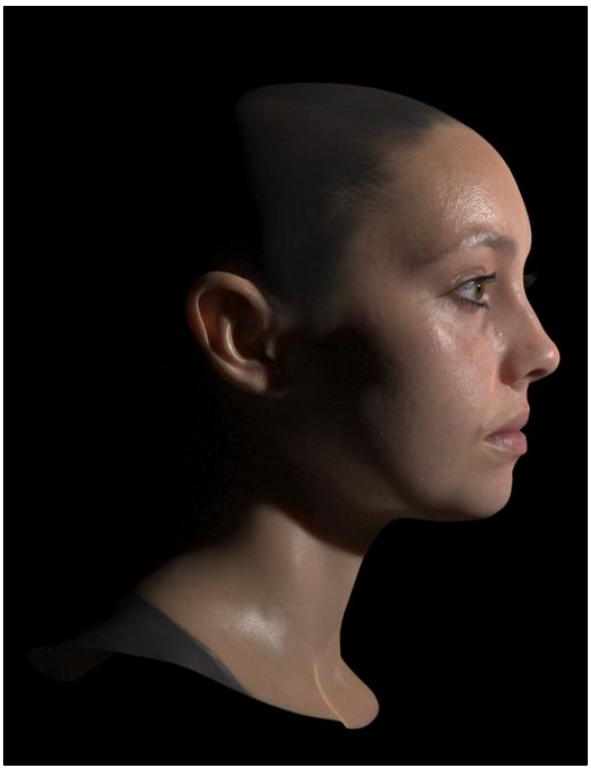
[39] Oculus. *Developer Centre.* Available at: https://developer.oculus.com/ (Accessed 18 May 2016) **URL**

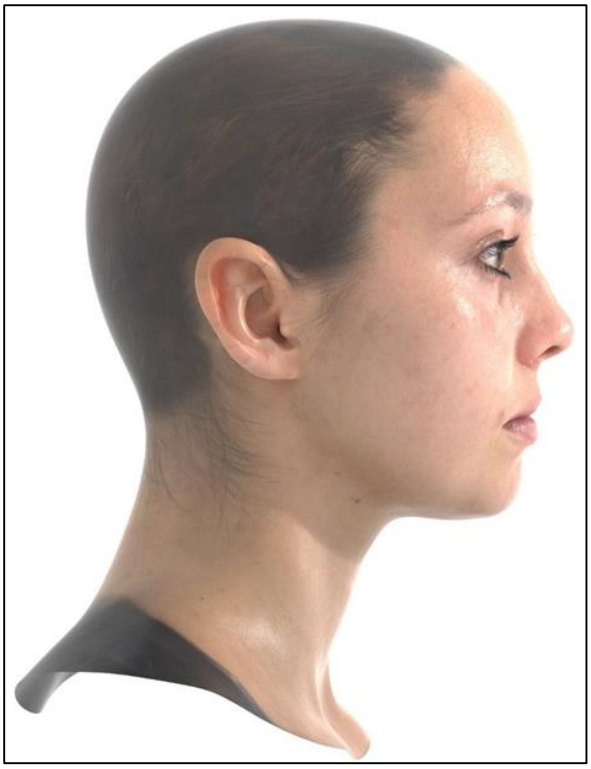
[40] Epic Games. *Blueprints Visual Scripting.* Available at: https://docs.unrealengine.com/latest/INT/Engine/Blueprints/ (Accessed 18 May 2016) **Documentation**

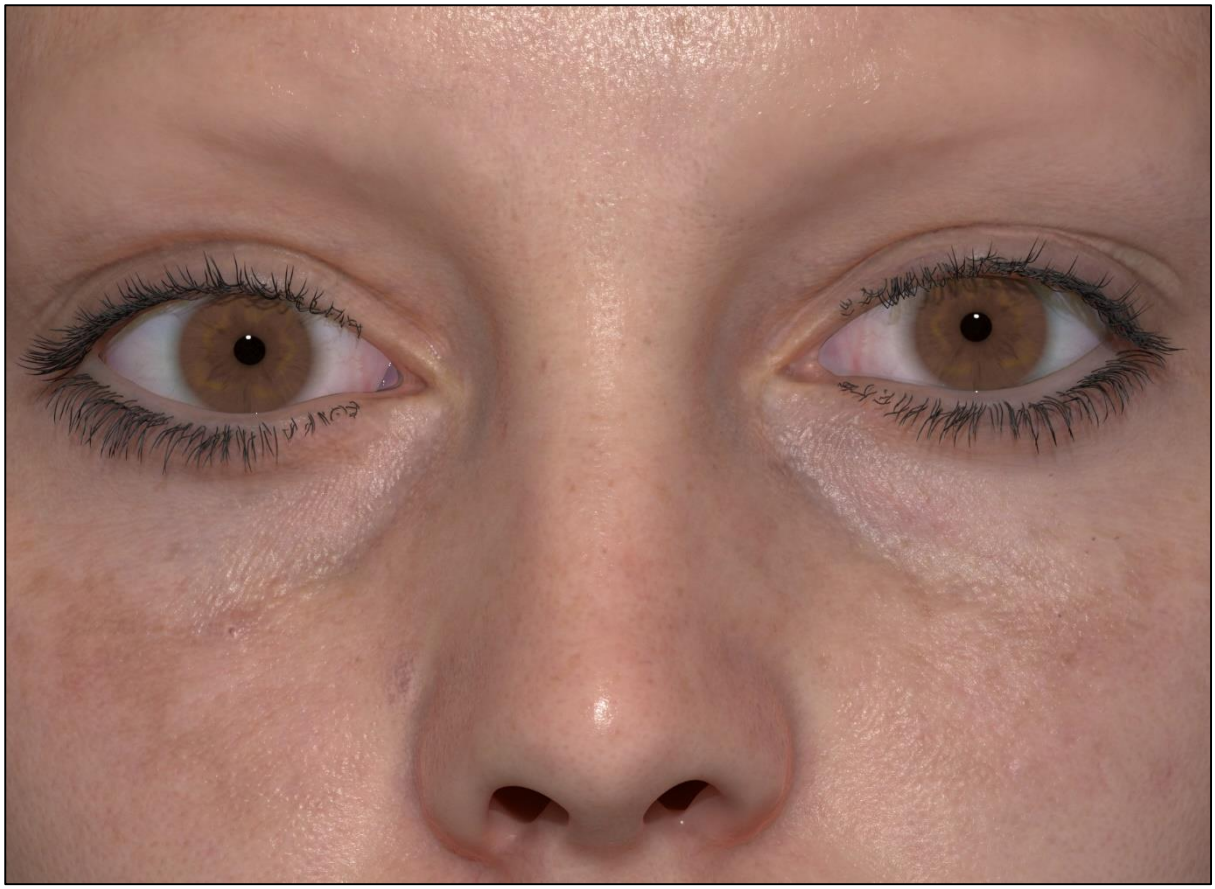
Appendix

Appendix A – Rendered Images

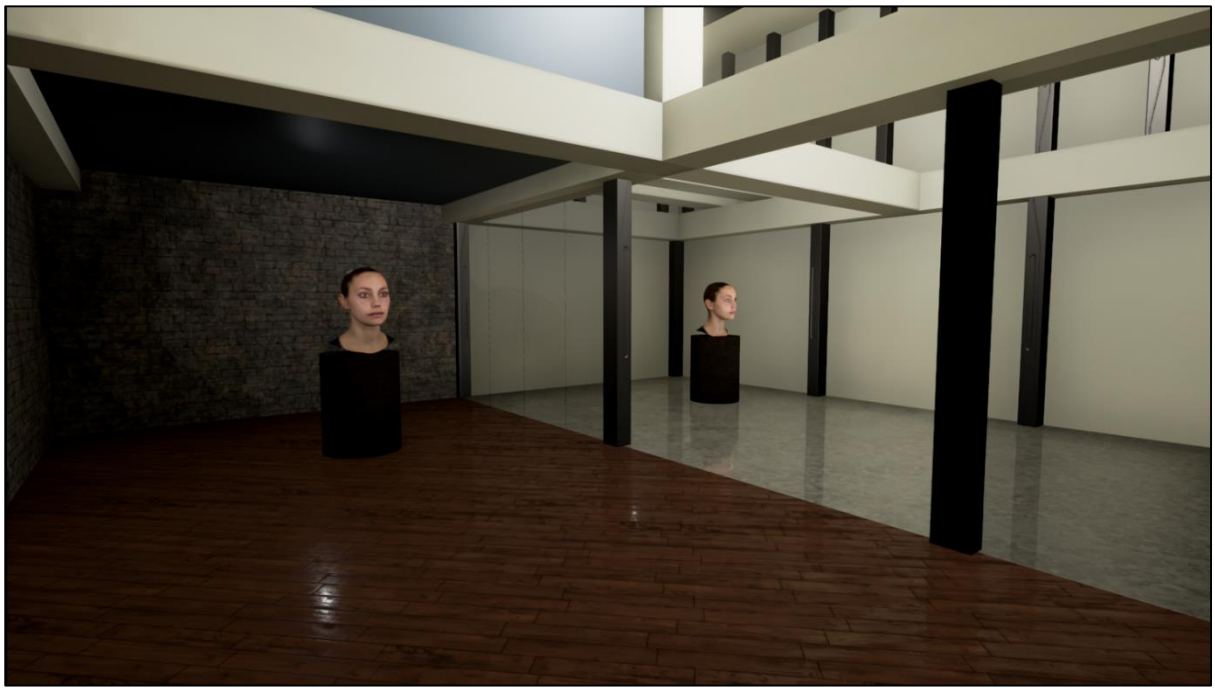
A.1 – 3dsMax Renders





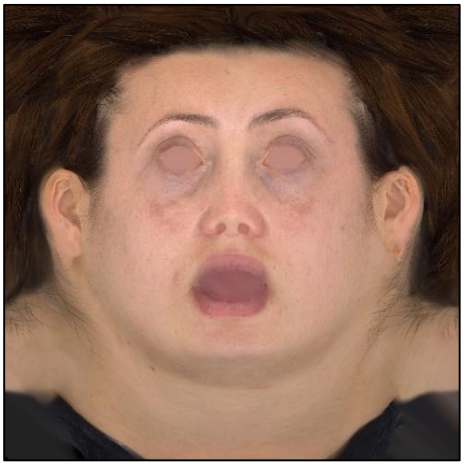


A.2 – Unreal Engine 4 Scene

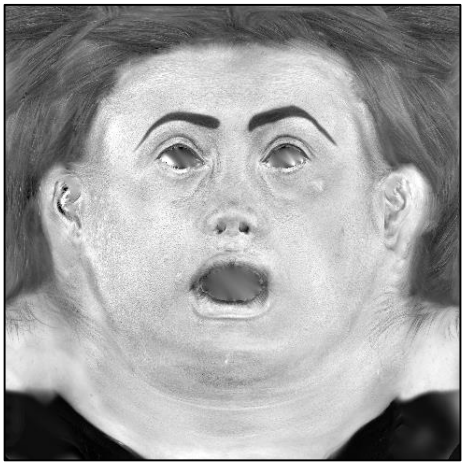


Appendix B - Textures

B.1 Emily Textures

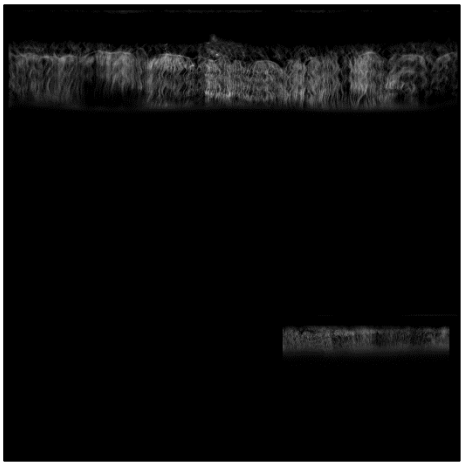


Diffuse Texture



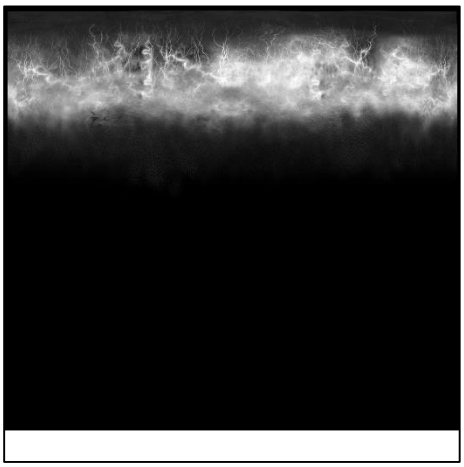
Specular Texture

Single Scatter Texture



Inner Eye Bump Map

Inner Eye Colour Texture

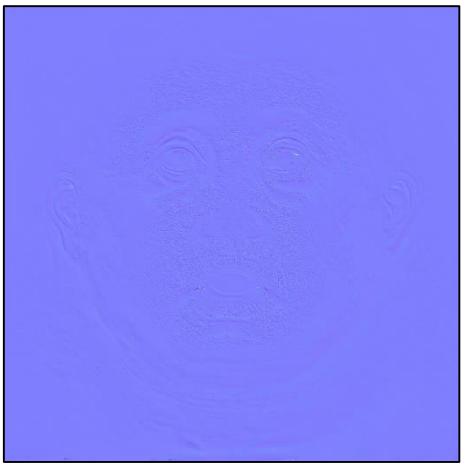


Outer Eye Bump Map

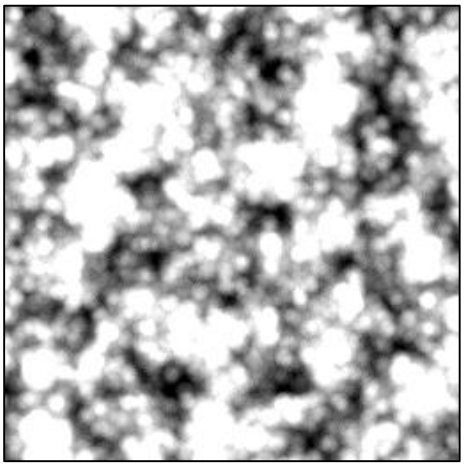
Outer Eye Colour Texture

B.2 Created Textures

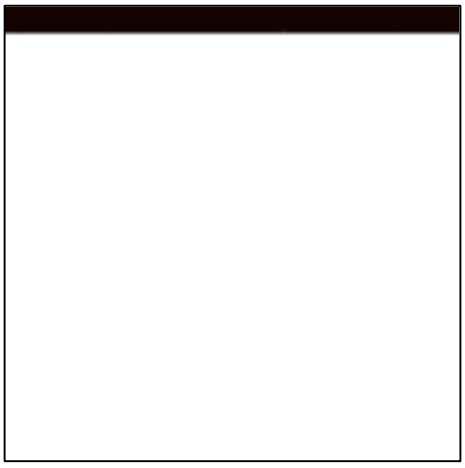
Opacity Mask



Normal Map



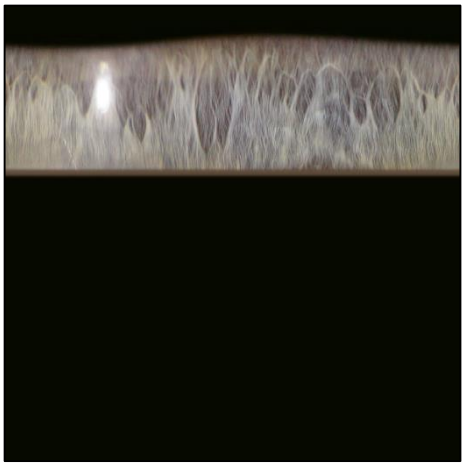
Plica Bump Map



Eye Mask

B.3 Other Textures

Sclera Colour Texture



Iris Ring Texture