View Dependent Real-Time Eyes for Virtual Characters

by

Daniel Wilson, BSc.

Dissertation

Presented to the

University of Dublin, Trinity College

in fulfillment

of the requirements

for the Degree of

Master of Science in Computer Science

University of Dublin, Trinity College

October 2012

Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

Daniel Wilson

August 29, 2012

Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.

Daniel Wilson

August 29, 2012

Acknowledgments

I would like to thank my supervisor, Rachel McDonell for her thoughts and kind advice throughout the duration of this project. I would also like to thank our course director John Dingliana for his hard work throughout the year, as well as the participants of the perception study for this paper. Finally I would like to thank Alana Sheridan for all of her support and most excellent tea making abilities!

Daniel Wilson

University of Dublin, Trinity College
October 2012

View Dependent Real-Time Eyes for Virtual Characters

Daniel Wilson
University of Dublin, Trinity College, 2012

Supervisor: Rachel McDonnell

The ability to construct photo-realistic virtual worlds and people imbued with motion, personality and full real-time interactivity still eludes modern games developers. Indeed hyper-realistic human characters remain a struggle on current computational hardware. Due to the complexity of a human's motion, the subtle facial cues and posture of creating inherent emotion, and detailed interactions with the physical world; virtual humans epitomise the art of real-time rendering.

Much work has been published on creating very realistic motion on a virtual interactive human, using motion capture tools or complex physics based locomotion. Similarly the physics behind thousands of strands of flowing hair can be accurately represented in real-time through established principles. Other complex human interactions with the physical world such as realistic shadows or the behaviour of light passing through multi-layered skin have not eluded works published on those topics either. Arguably

the final and perhaps most intricate feature of the human body that has yet to be robustly determined for real-time applications is the human eye.

The purpose of this paper is multi-dimensional. First, we answer why real-time eyes with photo-realistic fidelity are yet to be accomplished in any widespread manner. Next, we survey the state of the art regarding virtual eyes. A flexible material suite is presented for real-time eyes that accomplishes near photo-realism, and its efficacy is analysed with a perception experiment. Finally the significance of what is necessary to render the eye, and a robust manner in which this can be achieved is investigated.

The results of the novel, configurable shader system demonstrate the versatile effects that it can achieve. The importance of having various levels of detail with which to render the eyes is especially important for real-time or crowd based applications. The perception study presented in this paper validates the assumptions made for the shader system. The survey results on perceived realism correlate with the differing levels of detail used to represent the virtual eyes from various distances to the viewer.

Contents

List of Tables

List of Figures

Chapter 1

1.1 Introduction

Replicating human characters with real-world levels of fidelity in virtual environments encompasses a wide range of study in the art of computer graphics. The ability to do so is important for many reasons. The perception of the human body: its motion, volume, and interactions with the physical world are indeed subconsciously imbued within us from a very young age [89], [12] and [58]. The 'cloning' of people in virtual worlds which are imperceptible from their real world counterparts could in fact be regarded as the most decisive feat of graphics rendering. Applying those clones to interactive applications will undoubtedly lead to a sophisticated level of immersion and emotional attachment to the portrayal of characters in modern games.

Offline techniques for rendering people already provide photo-realistic imagery. To animate these characters adds yet another vast layer of complexity to the components of a successful human representation. The creation of hyper-realistic animations remains a cumbersome and expensive challenge limited to motion capture techniques which have a variety of restrictions. When motion is applied to a scene, physics based interactions must be accounted for which rely on the viewer, the lighting and the surrounding properties of the virtual world. Thus photo-realistic animations are currently difficult to achieve, and often take industrial render farms many hours to create each frame of animation.

As modern, accelerated hardware evolves at an exponential pace, consumer-level dedicated graphics processing units (GPUs) are now available with very high levels of parallel processing throughput. Near photo-realism is now achievable on fully animated characters at real-time frame rates (around 60 frames displaying per second). Real-time photo-realistic applications will undoubtedly supply the most immersive experiences for users, and are hence the focus of this paper.

Consequently, this paper pays special attention to the interactive rendering of one of the most complex perceptible features of the human body: the eye.

Figure 1.1: *A typical representation of real-time eyes in a modern game, taken from Electronic Art's Sims 3™ [93].*

Figure 1.2: *Offline eye rendering is already photo-realistic, as this Zbrush™ sculpture demonstrates [80]. There is currently a wide discrepancy between offline and 'game-ready' eyes.*

1.2 Motivation

There still remains a particularly wide discrepancy between offline and real-time results in human eye rendering. Figures 1.1 and 1.2 demonstrate the stark difference between both practices.

In the past few years, much attention has been paid to animating full human forms in a realistic manner, as this is the largest and most contributing factor of human to human perception. As discussed by Ennis [23], and Hertzmann [36], the gait of an animated character actually holds more significance than the realistic portrayal of many other features which define the character. Users subconsciously discern the realism of a virtual entity by witnessing their motions, even at long-range viewing distances.

Other extensive areas of academic study related to the representation of humans to date include complicated lighting models to represent how the *skin* is shaded. It is now possible to display the effect of light photons which enter the skin, scatter, attenuate and exit the skin at extremely close levels of detail in real-time. Perhaps the most realistic skin demonstration to date is discussed in Chapter 2.5 in further detail. Another complicated feature of virtual humans is hair rendering. Human hair,

particularly in motion, remains an intricate task due to the sheer quantities and physical properties involved in displaying interactive hair strands faithfully.

While a significant volume of literature exists on replicating the physical properties of human hair and skin, surprisingly little has been published regarding the final most complex and significant feature of the human body: the eyes. This is partially due to the fact they occupy very little relative space in the body, especially in comparison to the hair, skin and gait of a character. In fact graphical processing power has been almost exclusively reserved in the past for realising complex environments and lighting. GPU horsepower has been primarily set aside for coherently animating the millions of vertices on the characters that make up a typical gameplay scenario, and rendering large and obvious entities such as the motion of the body and hair.

Elaborate details visible through light interaction with the eyes are only distinguishable at relatively small distances even in the real world. This means that representing realistic eyes in a timely manner may only really be necessary for close-up facial imagery. At further distances they simply do not occupy enough pixels on the screen to warrant intricate lighting calculations. If there are many characters on the screen simultaneously as is the case in the majority of modern interactive titles, the required processing power increases at a linear rate, extending the problem particularly in heavily crowded scenarios. This paper explores that constraint by addressing some of the common eye components that can be withdrawn, specifying how and why they should be withdrawn from eye renderings at different resolutions of detail.

With an arsenal of simulatory capabilities, many modern real-time applications take a filmic approach to their themes and plot lines. It has therefore become more prevalent than ever to render human faces at close range, and without doubt the most significant feature of the human face in these instances is the eyes. This paper intends to address the realistic portrayal of eyes, but in a 'game-ready' fashion. Game-ready eyes must utilize the full power of a GPU but only when necessary. It is extremely wasteful to render eyes at the highest resolution of detail and instruction count when those instructions are not physically capable of displaying on screen, or being perceived by the user. Assumptions will therefore be examined on what exactly is necessary when rendering the eye at different distances and angles from the viewer. This is particularly important in computer games, where many animated characters may occupy the screen at any one time. Consequently, a game-ready eye must conform to 'level of detail' (LoD) constraints. This allows the programmer to define what components can be assigned, removed or adapted when rendering the eyes at different levels of detail.

1.2.1 Paper Layout

This report describes three essential steps to realistically represent eyes, and discusses an accompanying demonstration that is used to gather the drawn conclusions:

- The eyes must be as photo-realistic as possible, in order to maintain the best results perceived by users.
- The eyes must be game-ready: have LoD functionality, and satisfactorily render many eyes on the screen at any arbitrary distance from the viewer.
- Finally the assumptions for creating the eyes at different fidelities of detail must be validated through a perceptual study in order to ascertain the successes and flaws of the demonstration. This gives a valuable insight into the significance of how potential users comprehend the simulation.

Chapter 2 examines the current state of the art for rendering materials related to the final composite of the eye. Each quality is discussed, along with a brief overview of some of the key related technologies in use today. Chapter 3 deliberates the planning and design of the application, of the perception study and of the paper. This includes the reasons for the choice of platform, and the approach that was taken to formulating the code. Chapter 4 details the implementation of the shading algorithms for each quality of the eye, as well as their LoD functionality, and how significant problems were overcome. Chapter 5 analyses the performance of the programs and results of the perception study carried out. Chapter 6 reflects on the importance of LoD functionality. The results of the perception study are examined, as are precisely what details may be omitted from the eye rendering, and when it is necessary to do so.

Chapter 2

State of the Art

The following section includes a survey of related work and projects that pertain to realistic real-time eye rendering. Due to the broad and detailed nature of this topic, it has been divided into independent and relevant subjects. François et al. presented "Image-Based Modeling of the Human Eye" in 2009 [28]. This is a culmination of many of the following related works, and most importantly achieves the most in-depth and realistic representation of real-time virtual eyes to date. It is hence a primary focal point for this paper, and several of the topics relate to the work carried out by those authors.

2.1 Anatomy of the Eye

Extensive medical literature exists on the anatomy of the human eyeball, as it does for every part of the body. The eye however is a particularly complex organ with highly detailed tissues, as unique as a finger print. Studies of the chief components of the eye are outlined in this section.

Mann [50] presents a modern comprehensive paper on the composition of the entire eyeball, with particular attention paid to light entering and leaving the eye. Masters [51] presented an interesting computational study on the anatomy of the eye. It was reconstructed by taking many photographs at small degree increments across the axes of the eye, which were assembled to create a volume rendering of the eye for medical purposes. This is evidently not appropriate for any kind of real-time application, and the techniques have dated significantly. Much work has been done on the physics behind the rods and cones of the retina that receive and process light [38]. Work like this is largely irrelevant to this paper, whose aim is to model light interactions the viewer sees, not how light is processed. Medical literature also provides in-depth information on the eye structure as well. The most relevant resource to date is the assessment by François et al. [28] for real-time rendering purposes, who used the medical

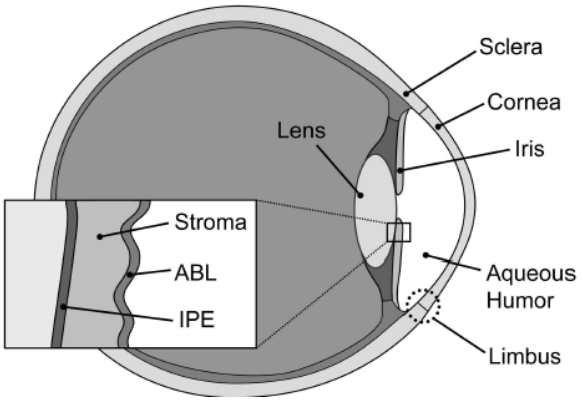


Figure 2.1: *A profile of the human eye [28].*

analysis carried out in [94], [85], [34], [39] and [21]. François et al. [28] note that Tuchin et al. [94] describe the scattering properties of the sclera. Saude [85] presented the diameter and index of refraction of the cornea, while Hecht and Zajac [34] note that light refraction mostly occurs at the outer corneal interface, as its refractive index is very close to that of the anterior chamber (the area between the iris and the cornea). Imesch et al. [39] note the importance of melanin in the chromatic characteristics of the iris, while Eagle [21] notes the scattering properties of different eye colours.

2.2 Capturing and Reconstruction

Capturing and reconstructing visually significant iridal features has been studied using several techniques. The iris of an eye is intricately multi-layered, and while replicating it by artistic means gives a great deal of freedom to the developer, it is too often highly perceptible, often leaving characters straddling the *uncanny valley*, noted by Brenton et al [12]. A developer has the option to create a high polygon mesh of the surface of an eye [62]. This option is largely undesirable: it is time consuming and requires an esoteric knowledge in order to produce a model with any reasonable degree of fidelity. With the advent of modern hardware, implementing custom shaders on a simple spherical model lends itself well to carrying out physically based rendering techniques due to the accelerated parallel nature of the hardware. François et al. [28] note the following four methods that attempt to reconstruct the human eye:

- Sagar et al. [84] used a simple Gouraud shading model and textures to represent and render the human eye for surgical practicing purposes.
- Lefohn et al. [47] present a method based on ocularist's studies: they introduce an aesthetically pleasant method for human iris cloning, based on multiple

hand-designed coloured layers overlaid to create convincing irides. Nevertheless, painting many semi-transparent layers is a time consuming task.

- Zuo and Schmid [106] generate human irides using synthetic fibers to mimic the fibrous nature of the iris layers. However, this method has been developed for the evaluation of iris recognition algorithms and does not address the problem of mimicking and rendering given irides, merely physically assessing them.

- Lam and Baranoski [102] propose a Monte-Carlo based rendering technique to accurately simulate the light scattering within the iris. This method uses the anatomical and biophysical properties of the iridal layers. Lam and Baranoski show that this rendering method can be used to accurately infer physical and chemical information from the human iris, such as the iridal melanin distribution, useful for medical purposes. However, this is an offline rendering method and does not address the problem of the human eye cloning.

Each of these methods does not specifically deal with the real-time rendering of irides in a realistic and flexible manner. François et al. [28] chose to instead clone the iris in a novel manner, by using macroscopic photographs. Once the iris image has been unrefracted and deconvolved, the remaining RGB image can be used to realistically approximate the surface. This is a largely automated process that minimizes user interaction, and results in a useful map that can apply multi-layered subsurface scattering to the surface. It is to date the most convincing real-time impression of the eye's complex surface structure.

2.3 Relief Mapping

In order to build an appropriate picture of the reliefs in the iris, a body of work known as relief texture mapping is adapted. It is important to note that it forms the basis for other related sections, and so is explored here. Oliveira et al. [71] first presented *Relief Texture Mapping*. This paper extends the very basic idea of texture mapping first presented in [14], in order to display varying levels of 3-D surface details and view motion parallax on polygons. Oliveira uses a two-pass approach: a relief texture is created: a texture extended with an orthogonal displacement (pre-warp function) for per-texel (texture element) depth. This pre-warp texture is then mapped onto a polygon in two steps: it is converted into an ordinary texture using a simple 1-D forward transform on the CPU, and then mapped in a conventional manner when it is sent to the GPU. An exact factorization of the 3-D image warping equation outlined in [54] is used as the pre-warping function to create the relief texture, and the depth information is computed as the distance from a reference plane to the sampled surface. Thus, correct views of geometrically rich objects can be obtained by rendering just a few

Figure 2.2: *A simple relief map gives the illusion of geometric micro-details [76].*

polygons that are relief texture mapped. Oliveira's relief texture mapping dissertation [70] provides a more comprehensive overview of the relief mapping technique that is presented in his published paper.

With the introduction of fragment processors, Policarpo et al. [78] extended and generalized the relief texture mapping technique for arbitrary polygonal models with an efficient implementation on the GPU. They achieved this by performing a ray-height-field intersection in 2-D texture space. The renderings present correct self-occlusions, interpenetrations, and shadows; and allow for other per-pixel lighting effects by exploiting the programmability of the GPU. This relief texture mapping technique makes use of texture filtering (e.g. bilinear), to guarantee that the height field surface will be continuous. As a result, extreme close-up views of the surface can be rendered without noticeable artifacts. This new relief texture representation is based on two depth layers instead of one. Dual depth relief textures use one more layer of depth to represent the back of the object. This significantly increases the quality of the effect as the object has tighter boundaries that are assessed with no extra storage cost and very small additional computational cost.

Oliveira and Policarpo generalized their relief mapping work once again to include the ability to represent non-height-field mesostructure details in [76]. This technique allows relief texture mapping to be carried out on multiple layers of a complex surface. The main contribution is the representation of several layers of depth and normals in the same domain and range (0 to 1). The ray-surface intersection procedure has also been adapted to suit an arbitrary number of layers. This situation is illustrated in Figure 2.4, where a relief texel may contain several pairs of depth and normal values.

One other interesting feature of this paper is the wide angular (but non-perpendicular) viewing range afforded to *relief impostors* due to the sampling of multiple depth and normal maps. A relief impostor uses a technique that utilizes modern shading hardware to perform ray casting into texture-defined volumes that give the illusion of a 3-D

Figure 2.3: *A relief impostor composed of a single quad, mapped with relief textures [76].*

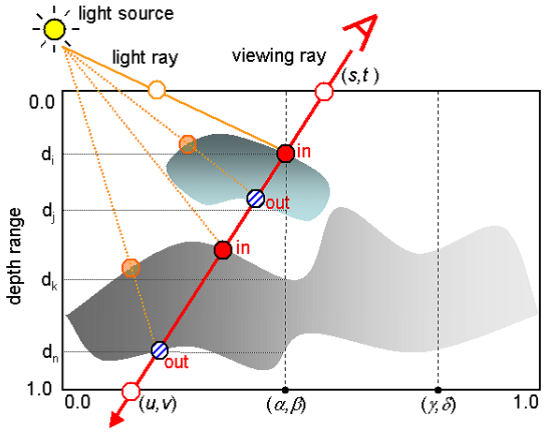


Figure 2.4: *A multi-layered relief texture map accounts for light interactions on multiple layers of a surface [76].*

model. Relief impostors are a resourceful means of affording detail to large swathes of entities at very little computational cost. The techniques described here to produce them are directly related to relief mapping, subsurface texture mapping and are an important LoD feature.

The *True Impostors* technique presented in [82] and published in [83] uses the pixel shader to ray-cast a view ray through the quad in texture-coordinate space to intersect the 3D model and compute the color at the intersection point. *True Impostors* support self-shadowing on models, reflections, and refractions, and it is an efficient method for finding distances through volumes. Andújar et al. [6] presented impostors with increased visual fidelity and rendering flexibility. These impostors analyse more maps in different angles. An algorithm determines the *"view-dependent selection of a minimal set of relief maps that maximise the visual quality of the final image"*, among other contributions such as generating construction and selection metrics for the maps. Figure 2.3 demonstrates a relief impostor from different viewing angles, generated from

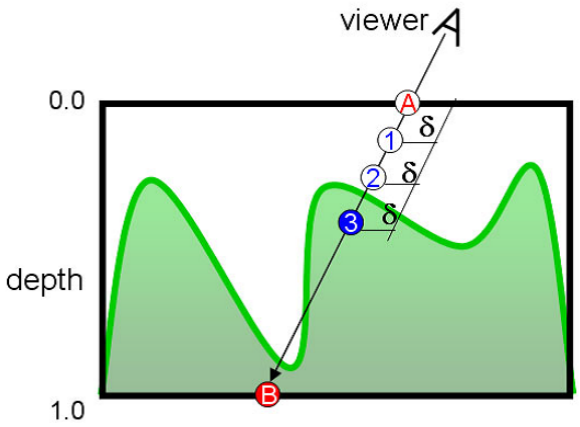


Figure 2.5: *In conventional relief mapping, a ray-marching algorithm samples a light ray δ times [77].*

just a single quadrilateral. Impostors are hence a valuable technique when rendering many low level-of-detail entities. While impostors are not used explicitly in this paper, many of the principles behind them are, and *True Impostors* would be a valuable extension for vast crowds of virtual characters.

Relaxed Cone Stepping for Relief Mapping presented in [77] uses a variation on the ray intersection algorithm depicted in Figure 2.5 where a binary then linear search is performed to find the view ray point of intersection. While the binary search converges very quickly, the linear search is prone to aliasing due to the step size δ taken. Cone stepping, first proposed in [43] essentially eliminates any artifacts by associating a circular cone (cone map) with each texel of the depth texture. The angle of each cone is the maximum angle that would not cause the cone to intersect with the height field (Figure 2.6).

Oliveira and Policarpo's relaxed cone stepping for relief mapping [77] epitomises their relief studies to date. It incorporates the better accuracy of the binary search while eliminating the aliasing effect of the linear search seen in conventional cone stepping. The constraint that a cone cannot pierce the surface is removed, and adapted: *"as a viewing ray travels inside a cone, it cannot pierce the relief more than once"*. By making the cones as large as possible (relaxing them) with this constraint, the technique converges to an intersection using a smaller number of steps. The relaxed cones eliminate the need for a linear search and consequently those associated artifacts. As the ray pierces the surface once, it is safe to proceed with the fast and more accurate binary search.

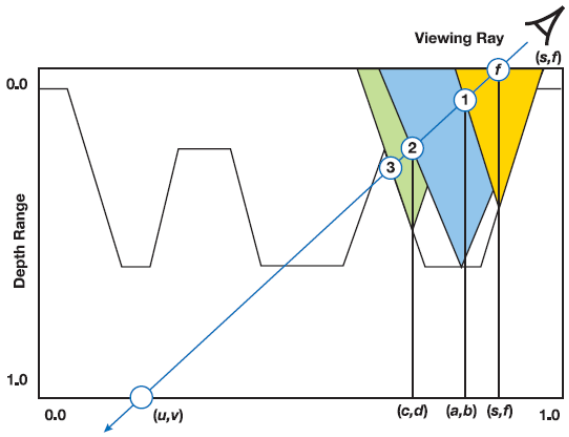


Figure 2.6: *At each pass of the iteration, the viewing ray is advanced along to its intersection with the cone centered on the current texel until the surface is hit. Relaxed cone stepping expands the cones to intersect once and only once [77].*

2.4 Subsurface Texture Mapping

The subsurface scattering of light is a broad physically based problem to which no one solution exists. Previous contributions in the area include techniques such as wrap lighting, normal blurring, texture space diffusion, and depth map techniques which are briefly discussed in this chapter, and detailed in [4]. Nishita [63] presented a range of physically based light scattering models for volumetric objects in natural scenes, such as clouds or water. Most shading models in real time applications consider the interaction of light only on the surface of an object. In the real world however, many objects are slightly translucent and light interacts beneath the surface. Common materials with a high presence of subsurface scattering include leaves, skin, marble, wax or milk. When light rays strike a surface, some photons are reflected away from the surface depending on its properties. In most cases at least some form light rays are not reflected. If they are not reflected, light enters the material instead, scatters some distance at least once (often many times) and may be slightly absorbed along the way before it leaves the surface. For real-time, certain assumptions and approximations must be made as the physically based radiative transport equation outlined in [16] is too complex for current hardware [33].

A simple technique that approximates scattering is wrap lighting. Wrap lighting modifies the diffuse function of the light so it wraps around an object where it would normally become dark. It is a crude approximation of Oren-Nayar's lighting model (which simulates rough matte surfaces [60]), [33]. Some improvement was made by Greene [33] to introduce a colour shift that accounts for the partial absorption of light, for example when rendering skin a red colour shift could be used [4]. This is still

Figure 2.7: *Advanced real-time subsurface scattering on skin. Interestingly, the developers may have noted that real-time eyes matching the photo-realistic nature of the skin in this demonstration is extremely difficult to achieve - hence the eyes remain closed [19].*

however a simplistic approach to rendering light interaction with complex surfaces.

Jensen et al. [41] presented an offline technique for rendering subsurface scattering as a diffusion process, with a spatial blurring effect. This technique blurs the normal maps of an object to account for some of the visual effects of multiple scattering. Perhaps the best approximation for real-time subsurface scattering to date has been carried out by d'Eon and Luebke [19]. This technique extends normal blurring, with texture space diffusion and complex filters that mimic the effect of a multi-layered heterogeneous structure. The effect requires many passes that blur irradiance maps, diffuse and normal maps and combines them to get an effective end result, depicted in Figure 2.7. Depth map techniques are useful for materials exhibiting large-scale scattering properties. However, they are not as useful when rendering the scattering properties of the iris as they account for the refraction of light as it passes through an object entirely.

Mertens et al. [55] and Wang et al. [99] present work adapting the [42] BSSRDF model (bidirectional subsurface scattering reflectance distribution function) that is dipole based. They precompute as much as possible to achieve higher frame rates, but these techniques deal with uniform materials only.

Donner and Jensen [20] extend these methods again, to include a multipole solution to subsurface scattering on many layers of material. It provides extremely realistic

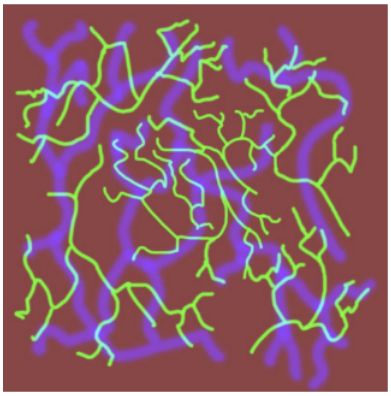


Figure 2.8: *The channels of a 2-D texture encode depth for the scattering of light [29].*

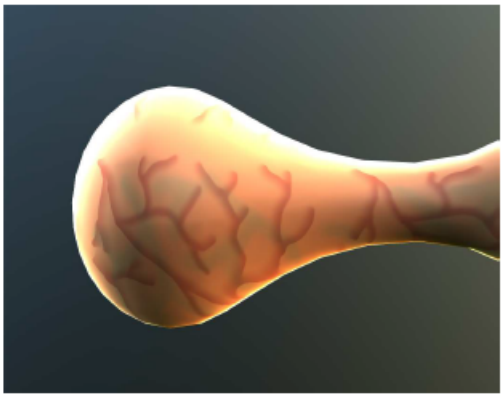


Figure 2.9: *A final subsurface texture mapped mesh, first presented by François et al. [29].*

results close to Monte-Carlo (the most rigorous photon transport available) estimations but not at interactive frame rates. Hegeman et al. [35] present a volume rendering subsurface scattering technique, but this does not cater for multi-layered materials either.

It should also be noted that a common method of representing complex or translucent structures in rendering is to use 3-D volume textures, where many (often hundreds) of 2-D slices of an object form the basis for its 3-D representation. This technique can accurately describe the inner composition of an object, such as the body and organs. One such implementation of GPU accelerated volume textures is presented in [45]. They can have a large memory footprint however, which may be a significant constraint for real-time rates. Furthermore, many surfaces (such as those related to the eye) only have subsurface scattering properties very close to the surface. In this case, they seem unnecessary and limiting.

The most useful technique pertaining to iris scattering properties is subsurface texture mapping, also presented by François et al. [29]. This work is a culmination of many of the techniques discussed, and primarily incorporates the work of Policarpo et al. [78] on relief mapping. Instead of representing the structure of a material above the surface of an object though, it uses the 2-D relief map to encode depth information for any incoming light interactions in the RGBA channels of the image. The red component is the upper layer and subsequent channels are lower layers of depth. This approach has a low computation cost with relative simplicity, and results in visually satisfactory results for complex translucent organic structures. It is therefore ideal for the complicated structure of the iris. Figure 2.9 depicts the end result of a simple map resulting in seemingly complex light behaviour beneath the surface of the object.

2.5 Reflection

The outer layer of the cornea is coated in a 'pre-corneal tear film', giving its surface a distinctive film of reflection. In [28], the eye is cloned for a real-time environment by first taking an iris photograph featuring these corneal reflections. The virtual eye is then aligned until its corneal reflections fit those in the photograph. The bidirectional reflectance distribution function (BRDF) is the function that defines how light is reflected upon meeting an opaque surface. Several examples of BRDF models exist. The popular Phong lighting model [75] is simple and highly efficient for real time applications. Cook-Torrance [17] is a more complex but flexible extension of Phong. These models are based partly on the physics of how light reflects off of surfaces, approximating unnecessary computations for the sake of throughput. Phong models all light sources as point lights, with each photon consisting of red, green and blue components. Phong offers two types of reflection: diffuse and specular. Diffuse reflection is light which is reflected evenly in all directions away from the surface, predominantly used in non-shiny surfaces. Specular reflection is reflected in a mirror like fashion, used on shinier surfaces. The light's *"angle of reflection is approximately equal to the angle of incidence"*. These reflections are the cause of specular highlights on surfaces [13]. Ambient light is modelled as light that arrives equally from all directions, and is intended to represent light that has spread around the environment through multiple reflections.

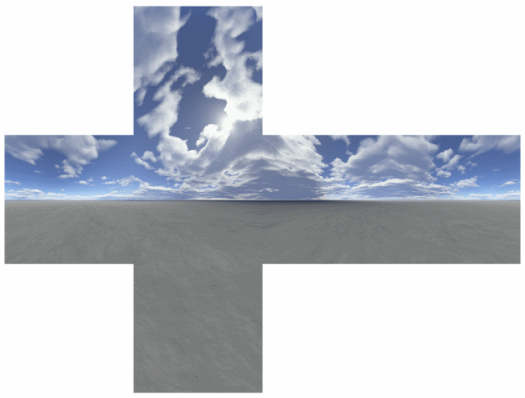


Figure 2.10: *A six face cube-map like this one is adequate to provide convincing reflections for a pseudo environment. Modern hardware can procedurally generate these every frame [24].*

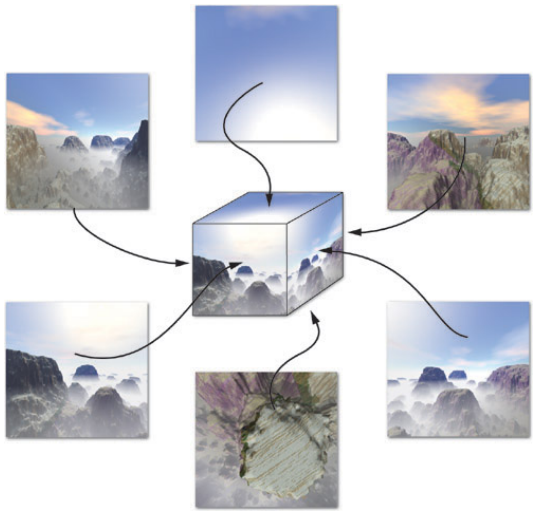


Figure 2.11: *Visualisation of the cube-map construction [27].*

Rendering basic specular reflection has hence changed very little since its inception. Reflecting the environment in real time is an important focus of eye reflection. Environment reflections were first proposed in [11], and Greene [32] extended the idea to use cube maps in a popular and flexible technique called environment mapping. The idea is to trace the reflected view vector back to some environment representation, such as a six face cube-map that might depict the sky (Figures 2.10, 2.11). The sampled cube-map colour is then applied to the output colour of the material. Cube-maps can be generated in real-time in a single pass of the geometry shader in the modern DirectX shader profile 4.0, and when coupled with normal maps and other lighting effects can produce a high fidelity representation of many different materials. Akenine and Möller [2] describe in more detail the process of generating glossy reflections from environment maps, particularly useful for reflections prevalent in irides if necessary.

One problem with cloning eyes from photographs as described in [28] is that the highly reflective cornea interferes with the quality of the image. Wang et al. [98] deal with this in a rather specific paper on removing reflections in human irides in order to gain a clearer representation. François et al. [28] note that this technique does not accurately recover the iridal colours that are frequently burnt out by strong reflections. In their paper they opt for polarizing filters on both the camera lens and the light in the environment to minimize reflections instead.

2.6 Refraction and Unrefraction

The appearance of the iris is augmented by the refractive qualities of the aqueous humor that fills the anterior chamber of the eye. It is considered to have a refractive index close to pure water, as outlined in [34]. In order to assess its true relief and pigment layout the light must be *unrefracted*. Refraction is the bending of light as it passes through one medium to another. In this case, light rays must change their angle of incidence when entering the cornea from air. Xavier [104] presents an efficient method for computing the refraction vector. Oliveira and Brauwers [72] present a real-time method of refracting light through animated, deformable models without the need for any preprocessing.

This is unnecessary for our eye rendering as the refractive components of the eye do not change shape. François et al. [28] therefore incorporate a pre-computed refraction and *Fresnel* function into their eye rendering system that efficiently simulates both by sampling a set of directions and points on the iris, and linearly interpolating between the results to build a complete picture of the refraction.

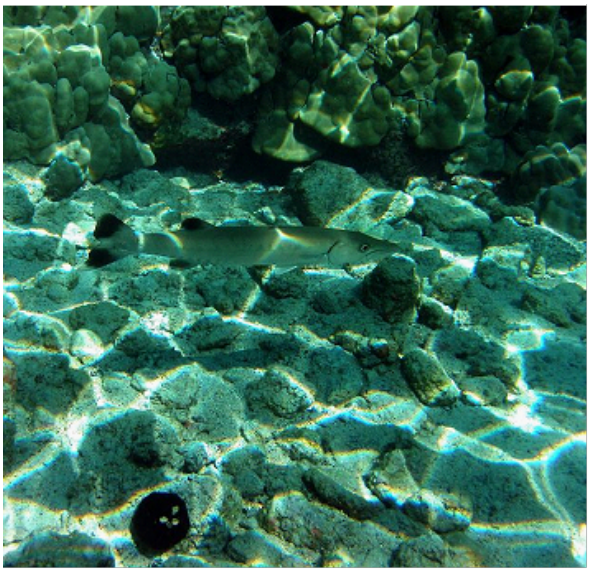


Figure 2.12: *The natural effect of light caustics through water is an ongoing research area in computer graphics [105].*

2.7 Caustics

Caustics are another important and complex field of research particularly for real-time rendering. Light caustics are the intricate processes in which light is focused via reflection and refraction through a curved surface or object. They can be readily witnessed as light passes through the inherent uneven surface of water in motion (Figure 2.12), or in the bottom of any mug. They are particularly relevant to light entering the eye at glancing angles. Two major classes of caustic rendering techniques exist: image space and object space [1]. Image space caustics such as Wyman [103] uses images to capture the effect of a caustic volume. Wyman presented a three pass technique, where the scene is rendered from the point of view of the light, a special depth map called a photon buffer tracks where refracted/reflected light has hit, and each location that has received light accumulates it into a caustic map. The caustic map is then projected onto the scene.

Ernst et al. [25] provide an object space technique in which a warped caustic volume represents how light converges or diverges from a region of the volume as a function of the distance from a caustic generating region. Converging caustic volumes in a region focus the light and result in higher intensities and caustic patterns.

François et al. [28] note that the caustics present in irides can be represented by a much simpler function similar in nature to their refraction function. This is an elegant solution to the complex problem. It samples a set of reflected/refracted light rays that hit a given point and returns a single direction, the result of incoming radiance with a

weighted sum of the caustic rays. By sampling many points on the iris, again they can be linearly interpolated to build a greater picture of the caustic behaviour in the iris.

2.8 Hardware Tessellation and Level of Detail

One of the primary contributions of this paper will be to achieve varying levels of realism when rendering irides for performance efficiency. Studies regarding the increment and decrement in levels of detail (LoD) in real-time environments are useful for a variety of reasons. Cebenoyan [15] notes that lower level detail models can have fewer polygons, lower quality textures or simplified shaders that still produce satisfying results. Automated geometric simplification (tessellation) of models is a technique used in every major game today. Linstrom et al. [48] provided a foundation of this work, which presents algorithms for tessellating terrain produced via a height field in a seamless manner. This has been extended as hardware has improved, to work such as Wiley [100] and De Smedt [52] who present more modern techniques combining the latest graphics APIs, the programmable geometry shader, vertex compression and adaptive displacement mapping.

Textures (including normal maps and others) are automatically generated and filtered in staggered levels of detail on modern graphics hardware, known as mip-maps. Ewins et al. [26] analyse how mip-maps should be chosen, and compares some selection and filtering algorithms that produce the best results. Today, most graphics hardware carries out this step automatically. Shaders alone can also be simplified depending on distance from the viewer, importance or other factors. Olano et al. [69] provide a description of techniques including geometric simplification and compiler optimizations that automatically reduces the complexity of an arbitrary shader.

Shader LoD can also involve simplifying lighting calculations by sampling fewer light rays for example, or streamlining the amount of per-pixel operations that must occur. Impostors are also an aforementioned impressive form of simplifying detail [49]. Modern graphics hardware utilising geometry shaders is capable of dynamically tessellating arbitrary objects, which is another significant area of study [92]. This is however a less relevant form of LoD for the purposes of this paper, due to the notion that the iridal detail presented here is maintained by effects such as subsurface texture mapping and not geometrical complexity.

The fact that all modern rendering engines support some form of automated LoD optimization demonstrates its significance. The cross platform Ogre^{TM}rendering engine [68] for example maintains an interesting assortment of optimization techniques that can be highly customised. Shaders can automatically choose which material scripts to implement based on factors such as distance from the camera or pixel count. Models are geometrically simplified and texture mip-maps are employed based on distance, among other items such as full impostor support or off-screen culling.

2.9 Analysing Shader Performance

One key feature of this paper is to maintain an interactive, ideally real-time (over 50) rate of frames per second. Nvidia™ [64] provide a useful guide on optimising shading algorithms for efficiency. Preisz and Garney [79] have several caveats in a chapter regarding GPU programming efficiency as well, such as how to find shader programming bottlenecks. Actual shader performance can be measured by useful tools designed for this purpose. NVidia's PerfHUD™is a tool for debugging and analysing Direct3D applications [65], while AMD's equivalent is GPU Shader Analyzer [5].

2.10 Crowds and Perception

Measuring visual fidelity is a broad field in itself. Studies have shown that humans have a keen and natural sense for distinguishing even offline computer generated characters from the real thing. Surman [89] and Brenton [12] build on and assess the 'uncanny valley' theory first hypothesized by Mori [58]. With the advent of hyper-realistic characters in real-time environments, developers increasingly strive to have audiences feel a sense of emotional attachment to characters. Much attention has been paid to rendering skin, hair and complex facial animations, while relatively little work has been conducted on rendering arguably the most complex and important feature of the face, the eye. Clearly realistic facial characteristics are the primary identifier for evoking emotion. McDonnell and Breidt [53] provide a correlated perception study on the trustworthiness of virtual characters, while Team Bondi's™ *LA Noire™* is a recent triple A title that used realistic motion capture and eye tracking to have the player detect lies. Quantic Dreams™produced *Heavy Rain*™, a critically acclaimed cinematic drama that has several very realistic characters. Anecdotally a fascinating real-time technical demo of human character performance is located at [18], which provides a short story centred on one of the most realistic real-time characters rendered to date (Figure 2.13).

The work of François et al. [28] is effectively cloning photographic images for real-time 3-D. Measuring the success of their eye work is therefore not perception based, but focuses on comparing luminance profiles of the rendered images and the photographs. They also use a root mean square method to determine the percentage of error between the R, G and B components of the rendering with the photograph. Their final result yields a maximum error of $\sim$5%, which is indistinguishable.

Eloquently transitioning between levels of detail adds another layer of complexity to the problem. The work of this dissertation is ideally crowd based at significant distances. Realistic crowds are important in many areas and again their shader based rendering is crucially understudied, with many solely focusing on animation and realistic or spontaneous AI for crowds. O'Sullivan and Dobbyn [73] note some useful means

Figure 2.13: *KARA is a robot who displays emotions in a tech demo by Quantic Dreams™. This complex demo is rendered in real-time on a Playstation 3, and achieves photorealism in many instances, particularly regarding the eyes.*

of optimizing all aspects of crowds. Elhelw et al. [22] have also provided work that tracks the viewer's eyes in order to define what features of photographs versus rendered images are most important in defining their realism, with the result that people focus on areas of high detail in particular.

2.11 Other Relevant Work

Finally, some notable projects relating to the work of rendering eyes, subsurface texture mapping and relief mapping have been carried out [67], [37], [9], and [86]. These either do not pertain to eyes specifically, are not real-time efforts, or result in non-exact clonings, as opposed to François et al. [28].

Chapter 3

Planning and Design

The following chapter will discuss the initial planning and development stage of the application and experimentation phases. These major design decisions impacted the evolution and execution of the project substantially. A brief discussion regarding an assortment of platform options is described, as well as the impact and caveats of the final decision. A description of the environment choices in which the final results were determined is presented. Finally, the methodology that was applied when writing the programs is briefly discussed.

3.1 Platform Choice

This project has been completed using a variety of tools. The primary aspect of the planning stage is the choice of *platform* from which to run the application. Several rendering engines are freely available and provide an open source license or similar for any products developed via those engines.

Microsoft's XNA™is a widely utilised C# based engine that uses DirectX™and their High Level Shading Language (HLSL). The Xbox Live Developer Network™ [57] community behind it is also very active. Several areas of contention are prevalent with XNA however: It is restricted to C# and HLSL only, commercial applications must be vetted through the Microsoft's EULA, and applications can only be developed for a limited number of items, namely the Xbox 360™, Windows™PC and Windows Phone 7™operating systems. One other problematic area is that at the time of writing, Microsoft's HLSL debugger (PIX™for Windows) is unavailable for Visual Studio 2010™users due to the migration of their DirectX SDK with Windows 8.

OpenGL™is a multi-platform graphics API that uses the Graphics Language (GLSL) for hardware shading [44]. Despite the large community that drives OpenGL, creating applications using it is arguably a more fragmented process in comparison to a fully-fledged rendering engine. Models, animation, texture loading, and menus are just a

few of the items which must be incorporated through separate libraries. While this leaves the developer with a large degree of freedom, it is inevitably a slower process to get a demo up and running. Irrlicht™ [40] is another free and open source rendering engine which was examined, and it supports both OpenGL and the DirectX specifications. However, many of the tutorials and forums for Irrlicht are not in English. The Unity™SDK [97] is a versatile game-making tool which although popular with a high degree of interoperability, does not provide access to much low level functionality such as the application-side processing of individual vertices on a model. The most prominent open source, low level cross platform rendering engine in use today is most likely Ogre3D™(Ogre). Ogre was developed by Steve Streeting in 2001 [68], and stands for Object-Orientated Graphics Rendering Engine. By assuming the role of a low-level rendering engine only, Ogre's core functionality was the ideal choice for this project. Ogre was chosen due to the active community forums, support for OpenGL, DirectX and hence the HLSL, GLSL and Cg shading languages. By using the C++ programming language, virtually any device that supports OpenGL or DirectX is able run an Ogre application based on the devices capabilities. Ogre has a rich built-in tool set that permits access to low level data, as well as much higher level functionality such as libraries for user interface design.

This higher level functionality means it is relatively fast to set up a simple demo with keyboard and mouse input, debug menus for displaying vital application data, and to process and display meshes using Ogre's proprietary *.mesh* format. It was also chosen to implement the hardware shading using NVidia's Cg specification. This is the third most popular shading language currently in use on modern programmable hardware. It is nearly syntactically identical to HLSL. Cg was introduced to provide cross platform support by targeting either an OpenGL or DirectX device. By using C++ and Cg in tandem with Ogre's rich framework, an adaptable and powerful application can be developed with relative ease, which extends the philosophy of this paper.

3.2 Creating the Environment

After setting up the rendering engine, it was decided how the environment should be planned. The experiments carried out are perceptual in nature, and the application has been designed with an object orientated approach so as to quickly allow eye models to be set up with varied parameters. The scene also relies heavily on modifying the eyes based on a level of detail (LoD) specification. LoD systems, discussed previously in Chapter 2.8, can be governed by several properties. Arguably the most important of these is the distance of the mesh from the virtual camera. Thus, a primary input method for this system is the ability to control the camera with familiar mouse and keyboard controls. Controlling the camera is also imperative in the development stages. Many of the effects cannot be witnessed without panning or rotating the camera about

the eye. By performing these motions, the user can truly examine the behaviour of light and its interactions with the iris and cornea based on the viewing ray.

A simple directional light was set up to illuminate any rendered meshes. Directional lights are lights without a position and are defined by a simple direction vector only. They are a simpler alternative to point lights which have a world-space position, or spotlights, which have additional attenuation parameters to create a cone of light which emanates from a point. A directional light is used when the light source is either infinitesimally far away (our sun is one example), or the surface being illuminated is very small. For our purposes therefore, a directional light is perfect. It also simplifies some of the shader calculations that may rely on a light position or fall-off value, because they have a constant intensity from all points in the supplied direction.

Finally in order to examine the importance of different forms of iridal reflectance, some animated models populate the scene. These are simple test meshes which have been purchased from the 3D modelling service Turbosquid [95]. In order to populate the scene with multiple models, several texture maps were created for the same model to help distinguish between them. The meshes were imported into the modelling package 3D Studio Max™, and rigged using the Character Studio™biped system within 3DS Max. Character Studio is a relatively fast tool for setting up simple animations on a skeletal structure. Motion capture files which are freely available from the Autodesk™site [7] were employed to add realistic walking and waving motion to the models. These motions provide an excellent source from which realistic reflections can be visualised. The environment is also populated by a large *sky box*. The sky box is a cube which encapsulates the entire scene, and is used to represent background details. A high resolution cube-map photo is seamlessly projected onto the corresponding faces of the sky box. This cube-map is also used in some of the reflection calculations discussed in Chapter 4.5.

The scene itself features a main character surrounded by a crowd of five other models. One character periodically walks past the main model, and another stands closely to the main character and waves. This set-up provides appropriate motion for some effects on the eyes of the main character, whose face is the focus of the study. The other characters stand to the side of the main character to form the basis of a crowd scene which is used for longer distance viewing ranges. Figure 3.1 shows the simple, complete demonstration scene from an isometric view.

The only other information displayed is a small frames-per-second and rendered triangle counter, which is necessary for the performance analysis discussed in Chapter 5.1. A conscious choice has been made to hide these counters from the user, as displayed numeric drops in the frame rate may influence decisions in the perception test.

Figure 3.1: *The application scene for demonstration purposes, the perception test subjects do not see the scene like this.*

3.3 Coding Approach and Debugging

This project features a relatively large assortment of complex shading designs. When it came to creating the shaders a coding approach with a gradual, rapid prototyping methodology was adopted. A development schedule was planned out for each of the main components of the eye. The creation of even very simple shading techniques was a valuable learning process, with many unexpected complications along the way. When problems arose, these were solved by consulting several resources. It was quickly understood how significant the Ogre 'log' file is. The log is generated by Ogre on every compilation of the application and shaders, and contains a wealth of diagnostics. System specifications are listed such as the supported shader profiles on the graphics card. Any asset loading is declared if successful, and errors and warnings for every file are generated.

The active Ogre community forums were also particularly useful, as were real-time rendering articles and books such as [3], [74] and [37]. The iris was first rendered using a simple Blinn-Phong shader, a small improvement on Phong shading discussed in Chapter 2.5, first published in [10]. This was gradually built upon to include caustics. Next, subsurface light attenuation and scattering which represents the complex folds and layers of the iris tissue is calculated. From there the light entering and exiting the cornea is reflected and refracted, which was appended to the subsurface and caustic techniques. The result is a logical division between the iris and corneal shaders, and they exist as two separate shader programs. Simplified versions of these two large shaders were then added for LoD experimentation, from the complex end result, to

extremely simplistic single colours with no reflection or refraction at all.

The shaders were written in two ways. The simple text editor Notepad++™was the first approach. It is a lightweight and convenient manner in which to quickly create a shader, particularly useful with a Cg syntax highlight plug-in. When significant problems arose that could not be solved using Ogre's auto-generated debug log, NVidia's FX Composer™ [66] shader debugger was deployed. This is a more substantial, fully fledged shader debugger. No *perfect* shader debugging tool has been developed to date, partly due to the sheer complexity of the millions of parallel instructions that execute on every vertex and fragment. FX Composer is currently (in version 2.5) a relatively stable shader debugger. It allows per-pixel debugging, stepping through fragment only instructions, and offers a visualisation window amongst other features. Shaders which are debugged in FX Composer must be in HLSL, but porting between Cg for Ogre and HLSL for DirectX merely requires swapping coordinate systems from left to right by transposing any basis transform matrices. Although syntactically identical to HLSL, Ogre's Cg uses a right-handed, column major rendering coordinate system. DirectX and HLSL use a left-handed, row major system. This means that any matrices used to transform vertices from one dimensionality to another in a shader must be transposed to result in the same calculations. For example, if performing Blinn-Phong shading in world space, Cg requires the position to be multiplied with the world matrix for the application to transform the vertex position from object to world space. Since matrix multiplication is non-commutative, HLSL requires the opposite for the same results. [46] and [59] provide a useful guides on the mathematics behind rendering coordinate spaces.

The design of the shader programs themselves has been demarcated by three issues:

- The Ogre framework defines each effect as a material. Materials are described in further detail in Chapter 4.2.1. Ogre material scripts define the look of a surface.
- The Cg specification divides vertex and fragment computations in the same manner as the other main shading languages. This division allows the user to specify calculations to be made at a less accurate but faster per-vertex rate, or at a fine per-pixel resolution. This is useful for LoD functionality.
- The materials must be defined in an adaptable manner. A wide series of materials is present in the project, where each one is dependent on the LoD requested by the mesh onto which it is applied.

3.4 Choosing Levels of Detail

It was decided that the LoD system in place should be governed by the same features to maintain consistency across each shader file. Thus there are four levels of detail in

total, each seven configurable effects. The levels range from low quality, to medium, high and 'ultra' levels. This represents a reasonable cross section of very low levels of detail for crowded, animated scenes viewed from a distance, to macroscopic views of a single eye. The levels of detail also represent techniques employed in modern computer games.

Actually choosing what should and should not be calculated at a particular LoD is one of the contributions of this paper. Hence, several combinations are presented in the perception study, and the results determine what is necessary at each of the four levels. Because every amalgamation of view distance, scene description and shader component LoD represents a combinatorial explosion of possibilities: at least 112 [1] for this arbitrary study, some clear assumptions have been made. The lowest levels of detail should either omit effects completely, or use basic colours as placeholders. High frame rates should be accessible at lower levels of detail even with crowds on-screen. The highest levels of detail should use high resolution, dynamic maps if possible, and require the greatest number of instructions in order to execute while still keeping real-time frame rates.

There are several methods in which an LoD strategy may be approached. It could be customised to account for the distance of the mesh from the viewer, or the on-screen pixel count of the mesh. Other aspects that may determine when LoDs are applied include the *amount* of meshes with the shader effect applied, resource intensive meshes such as tessellated and animated crowds, or GPU intensive complex particle effects or AI systems. For the purposes of this paper it was decided that the single most important factor which should affect the rendering of the eye is it's on-screen pixel count. There is nothing else particularly expensive in the scene, so the only other contributing factor may be distance. For this purpose distance would work well as eyes are of a consistent size. By considering the pixel count instead though, the eyes may be of any size and a logical LoD hierarchy will still execute based purely on the screen real-estate needed to render the eyes. This allows imaginative game developers to concoct games with eyes of varying screen-sizes with no further consideration necessary for LoD suitability.

[1] 4 levels of detail, each with 7 different effects (texture mapping, refraction, reflection, subsurface texture mapping, caustics, tessellation and other scleral effects) depicted by at least 4 predefined different sections of distance from the camera. This does not take into account using scenes with or without large or small crowds either, so the evaluation of every possibilty is not plausible for the purposes of this paper.

Chapter 4

Implementation

The execution of the shader algorithms is governed by two aspects: the conflict of *realism* versus *performance*. As discussed real-time computer graphics are on the verge of achieving photorealistic renders but it remains an immensely complex issue. For truly interactive performance, it is wasteful to calculate CPU/GPU instructions on details that are not visible. Polygons may be occluded from the viewer by other geometry. In which case, modern GPUs cull redundant pixels automatically, but this should be taken into consideration by the developer. Perhaps there is simply not enough screen real-estate to render such fine details, for example on mobile devices, or when the rendered objects are some far distance in the 3-D world from the viewer. This section will explore the factors considered when implementing a high degree of realism while maintaining real-time, smooth frame rates. The Cg shader programs are discussed along with the LoD system and the perception of the effects. Finally, some of the main problem areas, limitations and potential solutions are presented. All tests were carried out on a modern consumer grade machine with an Intel i-7 3930K hex-core processor, an NVidia 680-GTX GPU, and 16 gigabytes of RAM.

4.1 Shader Program Implementation

According to [8], the structure of the cornea has a symmetrical profile which can be algebraically defined by the expression:

$$pz^2 - 2Rz + x^2 + y^2 = 0,$$

where $p = 0.75$ and $R = -0.78$. This is hence the profile which defines the high resolution mesh used in the experiments. A programmatic mesh generation tool called *K3DSurf* [90] was used to initially define the mesh. This was then imported into 3D Studio Max 2012 (Max). Some final adjustments were made to the vertices, and it was attached to a base iris and sclera form. The entire mesh was unwrapped using Max's

UV tools for texturing purposes. The model was exported into an Ogre legible format (*.mesh*) using the OgreMax^{TM} [61] exporter plugin. Ogremax also features configurable tessellation settings which work in tandem with Ogre for the LoD strategy of the mesh. The correct UVs and tessellation requirements are embedded into the vertices using the *.mesh* format.

4.1.1 Ogre Materials

Materials in Ogre are defined in versatile material scripts. Material scripts support inheritance. They define one or more techniques for the material. A technique is identical to the HLSL implementation, and defines exactly what shader code will render the object that has been passed to it. If more than one technique is defined, Ogre allows the user to fall-back to another technique. Switching to a different technique might be necessary if the user's GPU does not support the shader model of an effect. Another important reason for switching techniques is when different materials should be used based on the LoD strategy of the application.

Each technique may comprise of many passes. A pass references the vertex and fragment shader code that will actually be executed on the mesh, providing a complete render of the object. Multiple passes can be combined, with different shader settings to create composite effects, such as deferred lighting.

Ogre provides additional LoD functionality within each material script, making it the perfect platform to quickly experiment perceptual LoD implementations. The `lod_strategy` field exists to be completely customised for application specific material switching. Another convenient method of creating subtly different materials is the use of *pragma* statements. A pragma statement is declared in the material definition section of the script, and essentially acts as a cheap conditional statement. Branching conditional statements and for-loops in shaders are notoriously expensive due to the parallel and voluminous nature of the calculations being performed. Pragma statements remove the need to include expensive conditional branches or the need to explicitly define a separate Cg shader. An example of a simple material script is presented in Appendix A. It demonstrates techniques, passes and material definitions, and features the use of a pragma statement which chooses whether or not to use a specular contribution. The following sections will overview the implementation details of each of the components in the demonstration.

4.2 Texture Mapping

Simple texture mapping is frequently the most important single effect on a mesh in a 3-D environment. Not only can diffuse colour be presented. Lighting highlights and shadows, textural clues to the material, and fine details can be represented with a

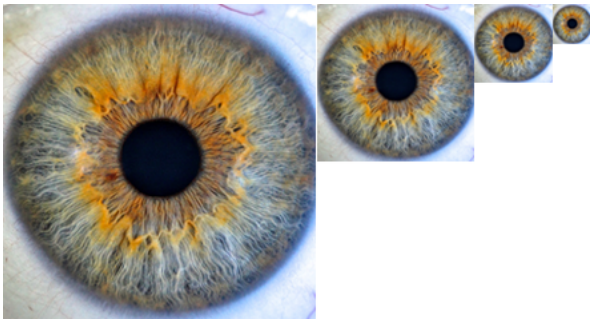


Figure 4.1: *A sample of the base diffuse texture mip-maps used on the iris.*

simple, well-crafted and high resolution map spread across the polygons of a mesh. For the eye, this detail is most noticeably in the iris. Consequently a high resolution iris photograph has been used as a base diffuse map, and the UV coordinates of the iris mesh match the layout of the photograph for correct texture mapping.

GPUs have dedicated hardware designed to read textures in a very fast manner [56]. A high resolution texture wastes processing throughput on the GPU if the details of that texture are not visible from the angle or position of the in-game camera. In this case, mip-maps (previously discussed in Chapter 2.8) can be automatically assigned instead to display the same texture map at lower levels of resolution as necessary. This is a well-established rendering trick, and subsequently Ogre supports mip-map generation and display. Hence the iris has four levels of map resolution, one for each level of detail.

4.3 Refraction

The implementation of real-time refraction is generally done in one of two ways:

- The first method is outlined in [87] and [81]. The effect is especially non-physics based, and fine-tuned by the artist. First the scene is rendered without the refractive mesh, and saved to a render target. This saves any colour information behind the mesh. In a perfectly non-refractive surface, this is how the image would look, with light entering the medium at the same angle it leaves. The light is perturbed by some form of noise map, such as a normal map with wave patterns for the surface of water. The perturbed ray is then used to sample the render target created in the first step. The result, depicted in Figure 4.2 is artistically pleasing. The normal map can be created by an artist for any purpose, and it may have motion, colour or any other composite influence applied to it for very convincing effects. The drawback to this method is that it requires the surface to be somewhat 'noisy', and a reliable normal map is necessary.

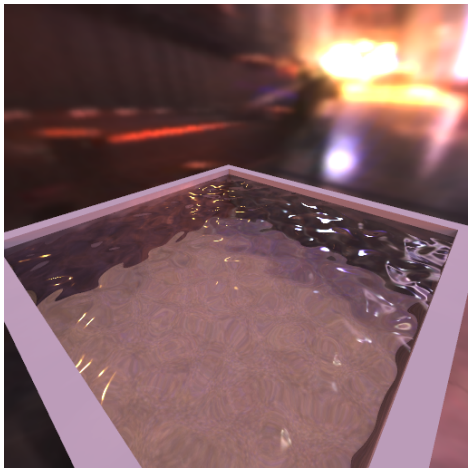


Figure 4.2: *An example of a non-physically based refraction technique [88]. The wave normal map features animated UV coordinates which provide a convincing refractive wave effect.*

Figure 4.3: *The physically based refraction depicted here samples a cube-map to simulate the perturbation of light rays. [27].*

- The second popular method is outlined in [27] and [101]. Here the incident light ray enters the surface, and is perturbed based on Snell's law of refraction [96]. This physically based law discovered by Ptolemy and rediscovered by Wilebrord Snellius in 1621 [31], defines a metric of light refraction between different surfaces. A perfectly non-refractive vacuum would have a refractive index of zero. The interface between air and water is approximately 1.333, and air/glass 1.06. This index is used to calculate the angle at which the light that enters the object should be perturbed. The perturbed ray is then used to look up a cube-map texture for the colour at the pixel where the viewing ray hits the surface. This technique is much more physically accurate, and suits the refraction of a smooth, non-noisy surface such as glass or the cornea. One disadvantage however is the reliance on a cube-map texture. Large dynamic cube-maps are still prohibitively expensive to create, and crucially static cube-maps do not represent any calculations that may occur at runtime beneath the refractive surface (post refraction).

The refraction created for this paper is thus a cross between these techniques, and hence somewhat more flexible. First a render target of the iris and pupil is created. The view/cornea intersection ray is perturbed based on the highly refractive index at the interface between the air and the cornea, defined in [34] as 1.378. The perturbed ray is then projected into tangent space in order to lookup the 2-D render target rather than a 3-D cube-map. The result is a refractive surface which can render any lighting

effects occurring beneath it as well, such as the subsurface-texture mapping of the iris. One drawback to this technique is that the render target is created at the start, before any refraction occurs. This means that any further calculations that rely on the light direction to create the render target in the first place are not using refracted light. One proposed solution would be to refract the light, store its new direction and create the render-target using the new light direction in an über-shader or a set of smaller shaders. The effect is still however convincing and physically based. In the low and medium levels of detail, refraction does not occur. This is because at these levels it is much harder to detect whether or not refraction occurs, as dissected further in Chapter 5.2.

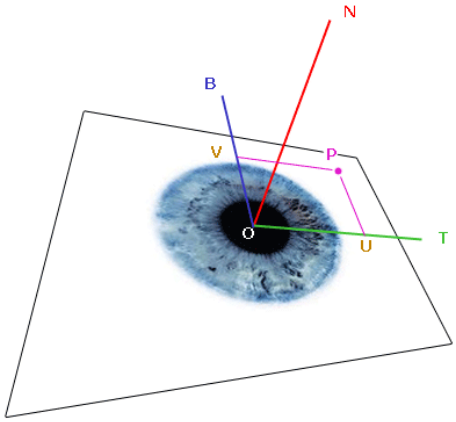


Figure 4.4: *The iris is represented by a plane. Perturbed light rays look-up the plane in tangent space to return appropriate pixel colours.*

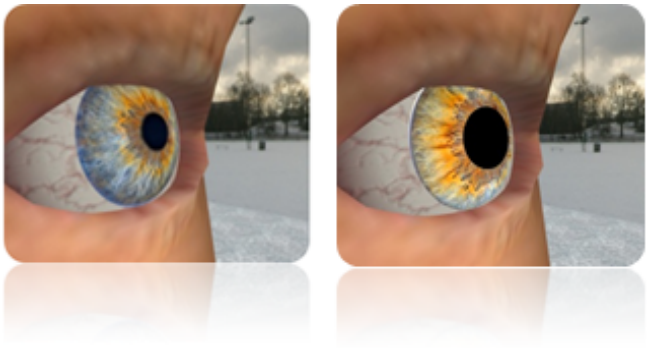


Figure 4.5: *Iridal refraction in the shader: left is the plain diffuse colour; right is the refracted iris giving more realistic magnified and perturbed features.*

4.4 Reflection

The calculation of correct reflections is of the utmost importance when attaining realism in a polished surface. Reflection is simulated in a variety of ways outlined in Chapter 2.5. There are two implementations in this paper, *static* and *dynamic*. François et al. do not indicate if they consider any kind of dynamic reflection [28]. Both techniques rely on calculating where the reflected light vector may travel, and how intense the reflected light may be. The reflected light is used to look up and display the values in a cube-map of the scene. In order to combine the effect of reflection and refraction, a *Fresnel* function is defined. The Fresnel function linearly interpolates between the amount of refractive and reflective colour based on the viewing angle to the object.

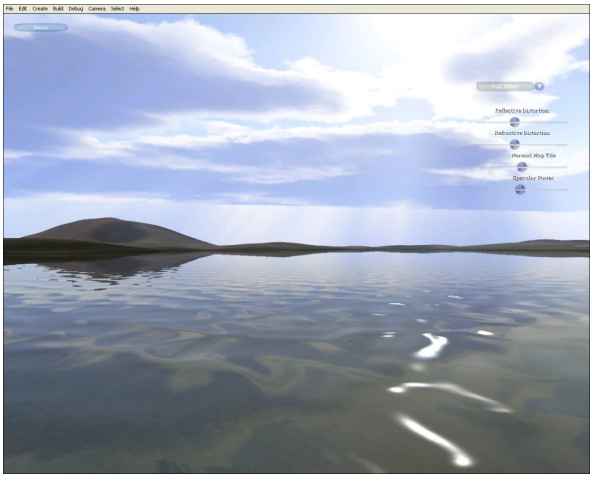


Figure 4.6: *A simulation of water that realistically depicts a mixture of pure reflection and pure refraction. The smaller the viewing angle is to the surface (hence further into the distance), the more reflection is displayed and the reverse holds true for refraction [91].*

This successfully emulates the real-world behaviour of refraction and reflection in an intuitive manner. Examples of the Fresnel correlation between reflection and refraction properties of water can be seen if Figure 4.6.

The use of a static cube-map means that the map is defined before run-time. The main disadvantage to this technique is that other objects in the scene, particularly objects in motion, are not present in the map. A variety of maps may be needed for different scenes as well. Importantly however, they are extremely efficient, easier to implement and are useful for a cheap form of low-resolution or 'blurry' reflection, such as that present on water. Alternatively, dynamic cube-maps are generated at run-time. Up until a few years ago dynamic cube-maps were strictly offline techniques only. With the advent of programmable graphics hardware, the six faces of a cube-map can be generated on a per-frame basis for many objects simultaneously on consumer level hardware. This means that the entire scene that is to be reflected can be accurately represented every frame.

The reflection of light on the cornea is an important aspect of depicting the eye with fidelity at close-range. However, these effects may be wasted at medium to long range LoDs in many cases, elabortated in Chapter 5 as one of the contributions of this paper. The decision of static or dynamic cube-maps should be made by balancing the processing power afforded by the scene, and assessing the perception of motion in the reflective surface for that scene. The dimensions of the cube-map may define an LoD too. A static, low resolution (for example 128x128 pixels-per-face), may be more than enough to add some colour to the chaotic nature of water tubulence. In contrast a

dynamic, medium to high resolution cube-map would represent a smooth, glass like surface at close range with a robust extent of fidelity.

The performance improves significantly when comparing a low-resolution static cube-map with a high-resolution dynamic map in the application for this paper, discussed in further detail in Chapter 5.

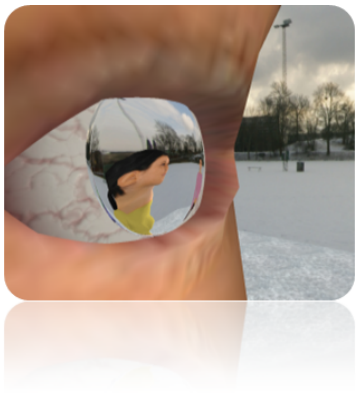


Figure 4.7: *A dynamic cube-map only is demonstrated on the iris, reflecting the animated characters in the scene.*

4.5 Subsurface Scattering

The most intricate single effect in the eye is the computation of light rays that meet with the iris. Some light is reflected using a fast and reliable Blinn-Phong lighting approximation. The remaining light is absorbed into the surface, based on a pre-defined subsurface texture map. As described in Chapter 2.5, this map encodes different levels of thickness and depth in the separate red, green, blue, and alpha channels of the texture. The major necessary steps of the implementation are described here.

Conventional lighting models trace the path of light photons *from* their source, back to the viewer. Light which has scattered through a translucent surface however, indirectly affects surrounding regions of the surface, from beneath it. In this case the reverse must occur in order to account for light from other displaced rays: the viewing ray must be traced back to the light source. The light from beneath the surface which lies on the viewing ray is assessed and attenuated by the properties of the subsurface map. Figure 4.8. depicts the two main components of the algorithm.

First the viewing ray is marched along in a number of sample steps. At each step, the intensity of the light at that point inside the surface is assessed:

1. The current layer where the sample point resides is calculated by looking up the subsurface texture.

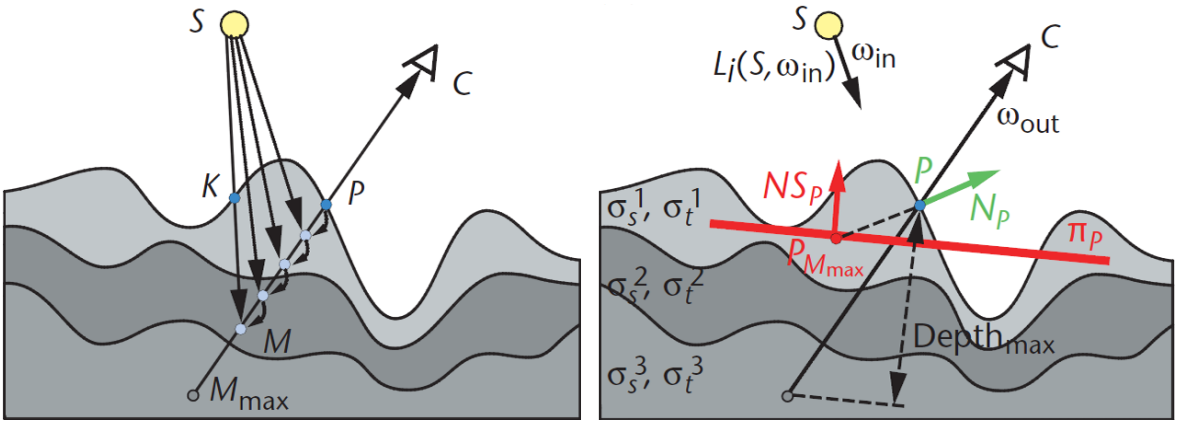


Figure 4.8: *Subsurface texture mapping ray marching algorithm variables. The resulting light intensity at point P is the summation of light at each sample point M. NSp is the approximated plane calculated in a pre-processing step [29].*

2. The intensity is attenuated by detecting how many layers the light has traversed through to get to the point, and the scattering properties of each layer is summated and taken into account.

3. This attenuated intensity is further reduced by assessing the amount of light which will travel back outwards from the sample point to the surface point along the viewing ray.

4. The total contribution of surface light is the sum of all sample points along the ray beneath the surface, up to a predefined maximum depth where light contributions are considered negligible.

Each layer has a set of scattering coefficients which affect the amount of each colour channel of light that may traverse through the layer. An example table of coefficients is outlined in Figure 4.9. One example would be light which passes through blood, in this case more red light photons would travel through than green or blue. In order to calculate the point of intersection between the light and the surface which becomes attenuated upon reaching the sample point along the viewing ray, an approximation technique is implemented. This approximation technique computes the plane *NSp* depicted in Figure 4.8. It is a rather complicated pre-processing step, which assigns a per-vertex approximated plane evaluated as a weighted average of the tangents within a certain neighbourhood radius of that vertex. The plane must be calculated for each vertex and passed into the shader for further calculations.

The luminance values of the iris are directly defined by its varying thickness and scattering properties. Hence darker regions of an iris image physically correspond to areas where light is more heavily attenuated. Therefore as stated in [28], the

Layers	$\sigma_s(mm^{-1})$ R	G	B	$\sigma_t(mm^{-1})$ R	G	B	g
Dermis	2.0	1.0	1.0	2.6	1.6	1.6	0.25
Blood	6.0	3.0	2.0	6.6	3.6	2.6	0.40
Organs	1.0	1.0	2.0	1.5	1.5	2.5	0.80

Figure 4.9: *A typical list of light scattering coefficients in organic tissue [29]. Note that a layer of blood will have a high coefficient for the red light channel because it is naturally closer to pure red.*

colour map of the iris is used as the subsurface map that describes the stromal, ABL and IPE layers of the eye (the foundations of the iris see Figure 2.2). Iridal freckles and imperfections are considered inconsequential concerning light attenuation for the purposes of this paper, notably unlike François et al. [28]. A slight ortho-radial Gaussian blur is also applied to the photograph due to the ABL layer of the iris being mostly composed of radial furrows. The resulting map used is depicted in Figure 4.10. Finally in order to account for the volumetric veins in the sclera of the eye, subsurface texture mapping could be employed here as well using a similar technique. From

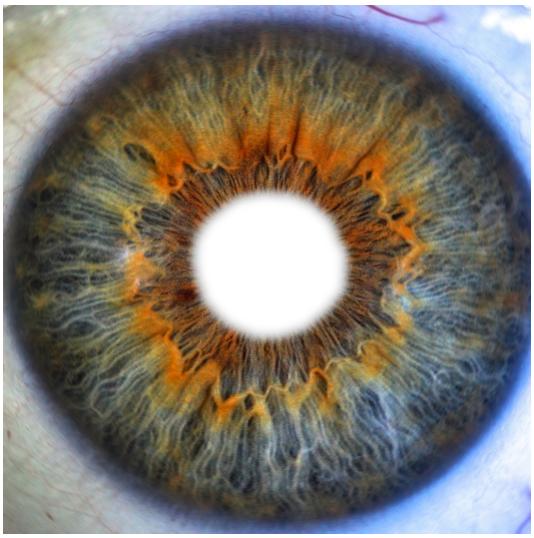


Figure 4.10: *The subsurface texture map used in the application is similar to the diffuse map due to the luminance values representing morphology thicknesses.*

a realism perspective, subsurface texture mapping is ideal to manage the transport

of light by balancing the efficiency of the ray marching technique with the GPU's streamlined ability to lookup texture maps. Non-real-time effects such as accurate Monte-Carlo light models provide very similar results to subsurface texture mapping, according to François et al. [28].

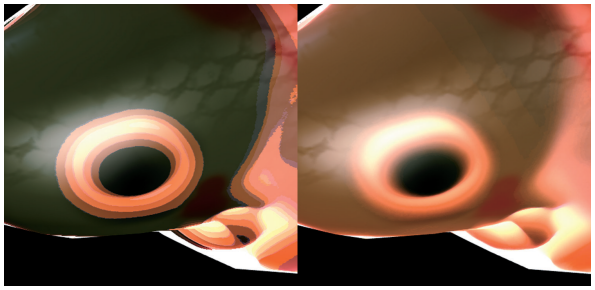


Figure 4.11: *This image from [29] displays a fish model with subsurface texture mapping using 10 samples (left), and 100 samples (right).*

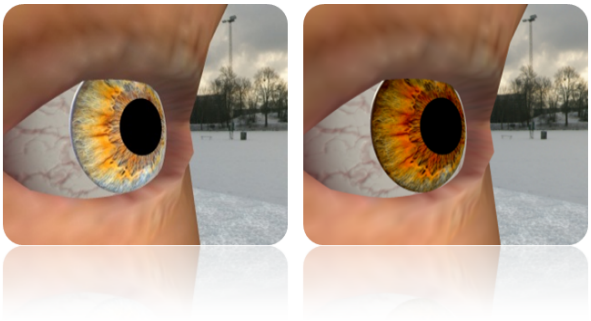


Figure 4.12: *The effect of subsurface texture mapping in the demonstration.*

The performance of the technique can be defined by several factors. The number of lights in the scene is not the main bottleneck of the effect. Using high resolution diffuse and subsurface maps will increase the fidelity of the image at a slight performance cost. The most significant factor though is the number of sample steps taken in the ray marching segment of the algorithm. A pictorial comparison is provided in Figure 4.11. Using only 10 steps for each pixel provides bands of colour where staggered attenuation occurs. Using 100 steps blurs the colours to produce much better results, but at a significant performance cost for clear reasons. For this paper, there is no subsurface texture mapping in the low and medium levels of detail. At the high and ultra LoDs, 20 samples and 200 samples are used respectively. For one eye on the test machine, the frame rate maintains an average of over 200 for the high LoD and drops to an average of 45 for the 'ultra' setting. This represents a significant performance cost, and was provided to give an insight between very low and very high flavours of the technique. An ideal average sample count could be around 50 for arbitrary scenes.

The effect is a resourceful and intelligent use of GPU hardware to approximate complicated structures such as the iris of the eye in a realistic manner.

4.6 Caustics

The caustics effect described in Chapter 2.7 has been simplified in comparison to more accurate techniques. Caustics are most prevalent on the iris when light enters the cornea at wide, glancing angles. The result lightens the output colour, and is de-

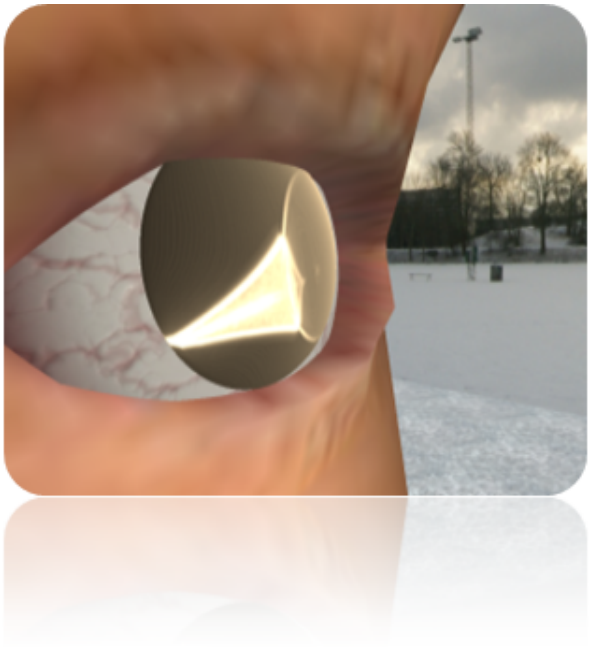


Figure 4.13: *The effect of caustics only in the demonstration.*

pendent on the main pigment colour of the iris and the colour of the light. For the implementation of relatively inexpensive but effective caustics in this paper, a caustics map has been applied. This map defines patterns which light caustics make in general terms. The map is summed with the final output colour of the iris based on the viewing angle and the light angle of rays which enter the eye. Figure 4.13 depicts the kind of effect achieved, and as light moves the caustics change shape and fade appropriately.

This method of caustic approximation is very economical in comparison to complex projective mapping techniques which sum light rays that intersect the same pixel. This would be implemented in a similar manner to how refraction is calculated, where the light ray intersection with the iridal plane is tracked. For the purposes of this paper it has not been implemented, as basic caustic mapping is convincing. Another manner in which the fidelity of the caustics may be increased would be to use a high resolution, complicated caustics map. Caustics are not present in the medium and low levels of detail in the application because the effect might only occupy a very small number of pixels. In the high and 'ultra' levels, a 256x256 and 1024x1024 resolution caustics map is used respectively.

4.7 Tessellation

Tessellating a surface increases the amount of triangles that define the surface. By adding more polygons, surface details can be represented at a much higher resolution for a performance cost. With more vertices to 'shade' through the rendering pipeline the performance cost may be very noticeable. If there are too few, meshes will simply not look good enough as no fine details can be represented.

Hence tessellation is a key feature built into all modern graphics APIs. The ability to blend between lower and higher polygonal representations of the details in a mesh dynamically can be specified in a strategy defined by the developer. This allows for flexible scenes with a high level of fidelity only when necessary, freeing up resources to render other in-game objects. This functionality is well suited for the purposes of this paper, where changing levels of detail is an important concern. The Ogre rendering engine has the ability to automatically tessellate models on command through various interpolation algorithms using different resolutions of mesh from the *.mesh* file.

In the application the eyes are tessellated only as a function of the distance they reside from the viewer, rather than pixel count. This is due to the tessellation algorithm taking into account the full size of the mesh which is inherently defined by its bounding box. Four levels of mesh resolution are exported from 3DS Max and encoded in the *.mesh* file. The different resolutions range from 150 triangles to over 8,000.

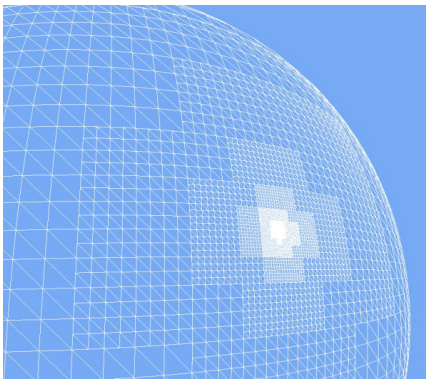


Figure 4.14: *Example tessellation. This dynamically occurs within the application.*

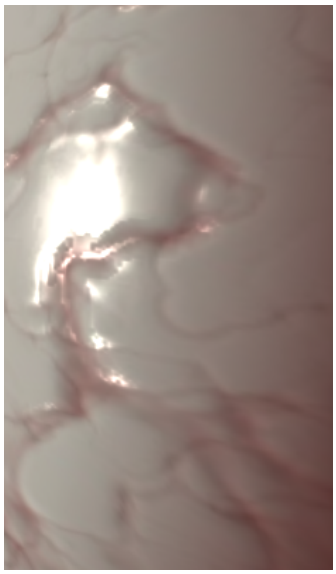


Figure 4.15: *The scleral detail includes Blinn-Phong normal mapping at macroscopic viewing distances.*

Figure 4.16: *The actual LoD results rendered at close range, from low (left) to 'ultra' (right).*

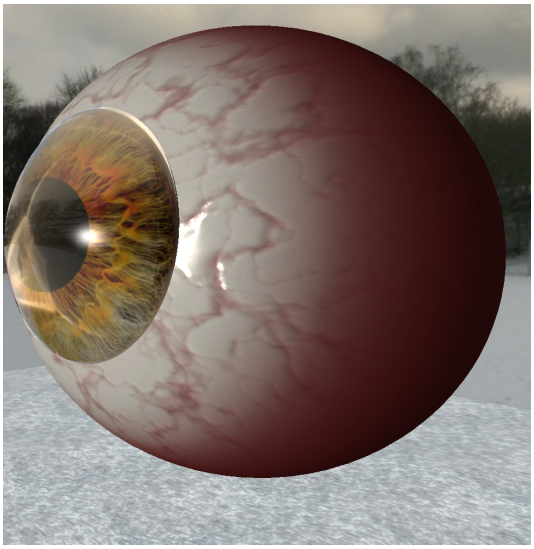


Figure 4.17: *The 'ultra' LoD.*

4.8 Other Important Effects

Some other effects complete the demonstration. The sclera and cornea of the eye both feature Blinn-Phong lighting which provides realistic specular highlights that represent the light source (in this case the directional light of the sun). Blinn-Phong shading is a well-established and very fast approximation of light source reflectance. Coupled with the other reflectance effects in use, the overall sense of reflectivity on the pre-corneal tear film on the eye is much more realistically defined for no noticeable computational cost. The small capillary blood vessels visible on the scleral surface of the eye are also present in several forms of detail. From very small diffuse textures without normal maps, to higher resolution diffuse and normal maps. This lessens the calculations required for long range LoDs. Even at macroscopic ranges though the scleral blood vessels should remain difficult to see, and the normal mapping effect is only subtly applied to give a thin layer of unevenness over the sclera.

Chapter 5

Analysis

Analysing the success of the application created for the purposes of this paper is a multi-dimensional problem. François et al. [28] focus on truly cloning iris photographs using an image-based technique. Therefore, the paper measures the success of that technique by comparing the precise pixel values of the photograph with the real-time renderings. One contribution of this paper is to analyse instead how various eye renderings are perceived and perhaps subconsciously assessed by the user. Hence, the perception study formulates the main governing metric for the success of the application. Several other factors are important for truly efficacious results and conclusions. As described, all eye renderings are to maintain a rate of 50-60 renderings per-second to be considered real-time. Furthermore, the lowest volume of instructions necessary to be indiscernible from higher levels of detail at varying distances is also assessed. Once again these intentions relate to the logical division of realism vs. performance. This section will discuss differing results and performance metrics related to these intentions, and the nature and outcomes of the perception study.

5.1 Application and Shader Performance Metrics

The application or CPU-bound processes that execute when running the demonstration are relatively lesser in comparison to the volume of GPU-bound shader instructions. In fact this application is almost entirely GPU-bound. One significant area of costly CPU bound instructions is the calculation of the approximated plane on a per-vertex basis, for each mesh which uses the subsurface texture mapping technique. As this is a pre-processing step however, it slows down the initialisation of the application by several seconds on the test machine, but does not hinder the application during the main rendering loop. Furthermore, the application does not fall below 60 frames per second unless more than 200 (the maximum) samples per-pixel are used in the subsurface texture mapping shader. In reality the sample count is a major contributing factor to

the frame rate of the program. Another important aspect is maintaining satisfactory eyes on crowds. The 'ultra' level of detail maintains a frame rate of around 50 with the maximum subsurface texture map sample count. This volume of samples is used to display one eye at very close range for demonstration purposes. The alternative levels of detail (LoD) maintain real-time frame-rates for many eyes on-screen. Indeed, the performance of the application is relatively constant for a multitude of characters: additional tests were performed using 30 (and thus 60 eyes) in a scene which sees a drop in frame rate to around 15 frames per second. One key factor in this case is that the LoD remains lower than the 'ultra' setting at all times due to the pixel-count LoD strategy in place.

Some extra performance tests have also been performed on the subsurface texture mapping shader, the most substantial in the suite. Using Nvidia's FX Composer we can see it consists of 137 instructions (11 texture look ups and 126 arithmetic calculations) as a minimum for every pixel: where there is at least 2 samples of the viewing ray. This is a considerable load on the GPU, and the solution implemented permits the use of subsurface texture mapping only at high and 'ultra' levels of detail.

Another substantial effect is the generation of dynamic cube-maps for reflection every frame. Using no subsurface scattering whatsoever still grants a frame rate boost of around 7-10 per-second when using large dynamic cube-maps versus smaller static counterparts. All other effects in the suite that define the eye have a lesser overall impact on the test machine by comparison. A factor of several frames is gained on the test machine by using the lowest to medium LoDs. One caveat is that the test machine uses a relatively powerful GPU (NVidia's GTX-680). Cards with slower clock speeds would realise more noticeable differences in frame rate, particularly when rendering crowds. Table 5.1 summarises the rate of frames per second rendered using one or two eyes depending on the LoD. The very lowest form of rendering: a basic mesh with a very small texture applied grants a frame rate of around 1,000. Again this is due to the speed of the GPU, and the rendering loop limit constrained by Ogre. Increases in fidelity, as expected provide decreases in performance, up to a point where rendering one eye at the highest level grants around 50 frames per second.

5.2 Shader Results and Perception Study

Table 5.1 defines precisely how each effect in the suite is maintained for every level of detail. The context for the decisions regarding the manner in which each level of detail should be rendered is driven by the perception study. Thus, the study is partially designed to see what a designer may remove and when, when tasked with rendering eyes in-game. The study itself has been conducted on a small group of 10 people. The group consists of 6 colleagues (with a keen sense for graphical discrepancies), and 4

Table 5.1: *Application Level of Detail Functionality*

Level of Detail	Low	Medium	High	Ultra
Polygon Count	∼150	∼500	∼2000	∼8000
Diffuse Map	128x128	512x512	1024x1024	2048x2048
Refraction	None	None	Physically Based	Physically Based
Reflection	None	Blinn-Phong with a 128x128 static cube-map	1024x1024 static cube-map	2048x2048 dynamic cube-map
Subsurface Map	None	None	10 samples along viewing ray	200 samples along viewing ray
Caustics Map	None	None	256x256	1024x1024
Scleral Effects	56x56 diffuse map	512x512 diffuse map	Blinn-Phong, 28x28 normal map	Blinn-Phong, 512x512 normal map
Frame Rate	∼999	∼700	∼200	∼50

individuals who have not played any real-time applications within the last 6 months. Having a sample of 'non-gamers' provides a more defined insight into the perception of realism in the images.

A collection of 7 different high definition video clips was played for each participant. Each video features two eyes rendered side by side at the same viewing distance from the camera. The arbitrary distances were selected to supposedly correlate with the LoDs: far, medium, close and macroscopic ranges of sight. This study aimed to verify whether the correlation holds true for the assumptions made regarding the LoD decisions. Both eyes are rendered using a cross section of effects, and Table 5.2 describes the combination of distances and LoD in each video. Figure 5.1 depicts a screen capture from each video. Each clip also features a rotating camera which orbits a 120° field of rotation about the 'up' axis of the character's face. This is particularly important because none of the effects can be correctly realised without witnessing the behaviour of light from different angles while in motion. It is also rather suitable because all realistic game environments feature some form of animated viewing camera. As previously described, the scene has been set-up with animated (waving, walking) characters to present reflected motion in the eye. This is useful for discovering the impact or lack thereof for the dynamic reflection effect, which displays the models in their animated form in the reflections on the corneal surface.

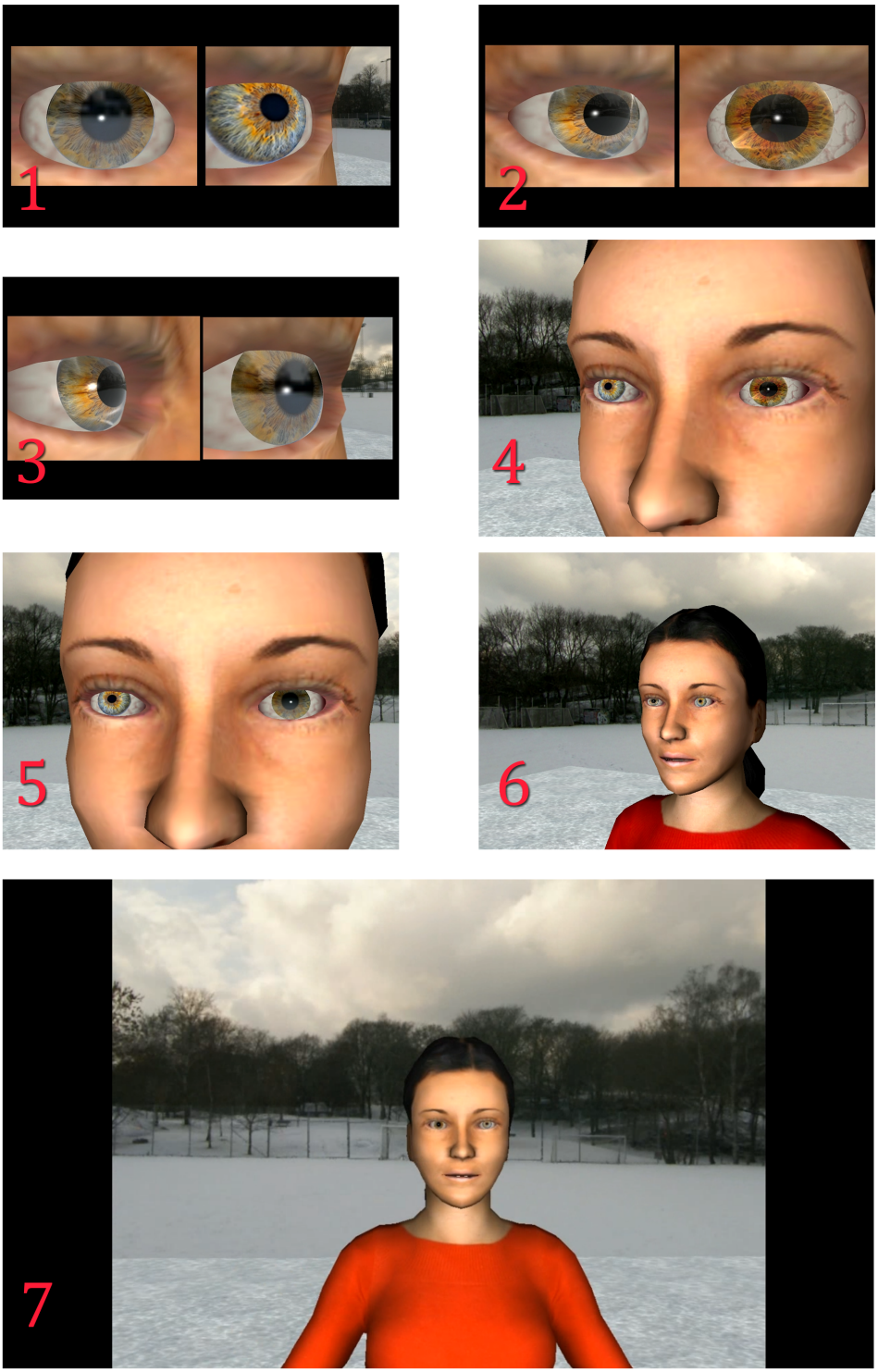


Figure 5.1: *A collection of stills from each of the 7 cinematics in the perception study.*

Table 5.2: *Actual Perception Experiment Levels of Detail*

Video Clip #	Distance From Eyes	Left Eye LoD	Right Eye LoD
1	Macroscopic	Medium	Low
2	Macroscopic	High	Ultra
3	Macroscopic	High	Medium
4	Near	Low	Ultra
5	Near	Low	Medium
6	Near	Ultra	Low
7	Medium to Far	Ultra	Low

On watching each cinematic a number of times, a brief questionnaire (presented in Appendix B) was supplied. If the eyes were distinguishable it was determined whether they were more or less realistic than each other to the viewer, and a realism rating was requested. The realism rating is a generic scale ranging from 1 (completely unrealistic) to 5 (near photo-realistic). The results for each eye, in every video are formulated in graph 5.2. This graph depicts the average perceived realism for the cross section of LoDs presented in the collection of videos. A major contributing factor is also the distance the camera had been placed from the eyes. The chart demonstrates the viewing distances for the cross section of videos also.

This graph provides at a glance an insight of how the realism has been perceived, and whether the camera's distance was a major contributing factor. In the experiment the videos were randomly demonstrated at the varying distances.

5.2.1 Perception Evaluation

Generally we can see that the actual and perceived levels of detail are proportionate, but heavily modulated by the camera's distance. Figure 5.2 displays an appropriate increase for the *perceived* LoDs, directly related to the *actual* eye LoDs. As the camera increases in distance, the perceived level of realism becomes less accurate in comparison to the assumed levels of detail. At the furthest distance from the eyes (in video clip 7), the lowest level is displayed (along with the ultra LoD), and the subjects awarded a much higher average realism rating of 2.5, in comparison to other ratings of the lowest LoD. This implies the effectiveness of the lowest LoD is proportionate to the viewing distances.

For closer distances, the actual and perceived levels of realism are relatively proportionate to one-another. As described, the study was conducted on a cross section of viewing distances and LoDs only because every combination would become convoluted for the testers. For example, the furthest distance had only the 'ultra' and low levels examined because these provide the most significant demonstration of what constitutes

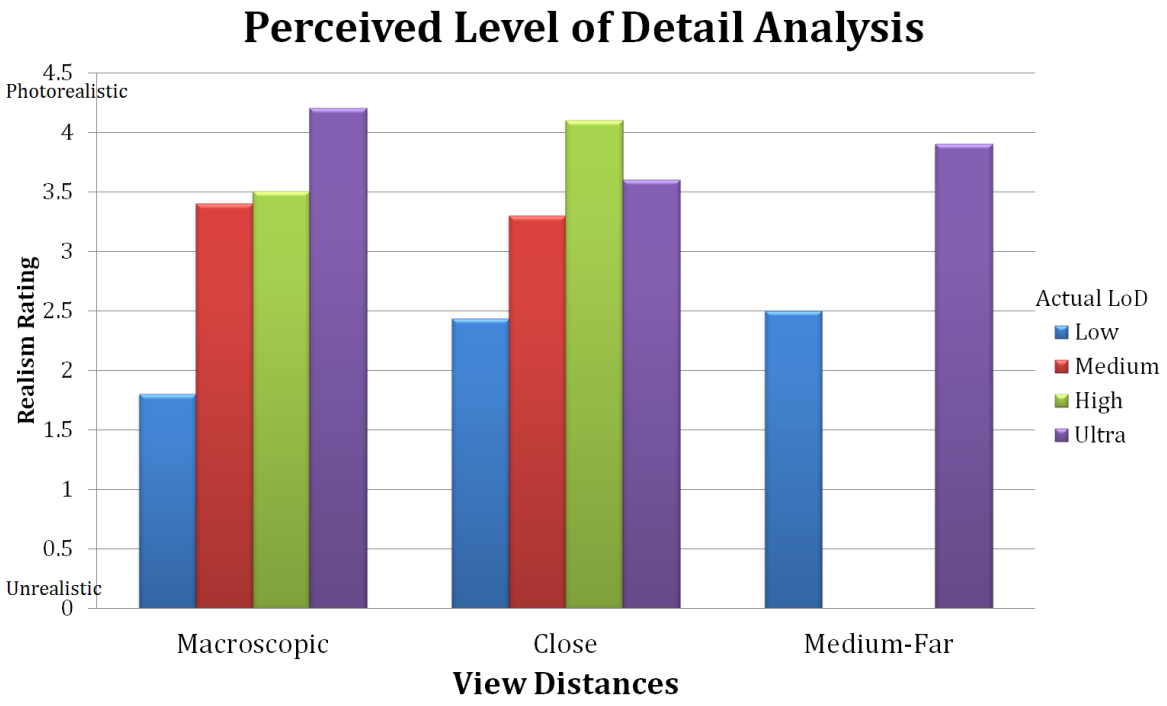


Figure 5.2: *For each video clip, the average perceived realism results for each rendered LoD is analysed.*

realism at these viewing ranges. Every combination was tested for the macroscopic viewing range because this provides the easiest view for users to spot any differences. Another notable trait is the perceived resemblance of the medium and high ranges of LoD at the macroscopic distance. This implies the differences chosen between these LoDs is suitable even for very close views.

5.3 Feedback and Comparisons

Finally some discussion pertaining to the comments received by users during the experiment is useful to give an insight into their choices. It was noted by several users that witnessing both eyes and comparing them to one another was slightly off-putting. As visible in Figure 5.1, the physical differences between the lowest and 'ultra' levels of realism is rather stark. As refraction occurs, the pupil is magnified. The iris takes on a more detailed, darkened hue from the subsurface scattering, and the addition of different forms of reflection is also very prevalent. These factors often result in two strikingly different eyes in the same character which in essence makes judging both more complex. The purpose of having two different eyes was primarily to see if users

might notice the difference, but a number of effects are particularly conspicuous even at medium to long ranges. A solution to this would be to double the amount of video clips by having the same two eye LoDs depicted in each video.

The most substantial effect noted amongst testers for the eye is the reflection. It was noted by 8 users that even simple Blinn-Phong shading adds volume to the perception of the eye. Having the shape of the eye react to the light by glistening a specular highlight dramatically increases realism for a very low computational cost. Furthermore, the dynamic reflection was only witnessed at the macroscopic viewer distance from the eye. As discussed this is the second most expensive effect, and by removing it (one of the key differences between 'ultra' and high levels), results indicate that only 2 users (which were colleagues) seemed to notice. Medium resolution cube-maps are important at close to medium viewing ranges, as the low resolution cube-map was disorientating to several subjects. It was noted that the low-level cube-map looked especially unrealistic, making the eye seem 'see-through'. It was noticed that the lower resolution caustics map on the high LoD looked unrealistic, and one user felt the high resolution caustics were unrealistic too. The tessellation was seemingly undetected by anyone, and the mip-mapped diffuse textures were successful too. About half of testers specifically noted the scleral normal mapping effects at macroscopic ranges of detail, and when higher resolution maps were used. The refraction has a large impact on the view size of the pupil and the folds of detail in the iris. Nearly everyone noticed when the eye was refracted and when it was not.

Chapter 6

Conclusion

The results of this paper are scrutinized in more detail over the following section. Realising truly useful outcomes through the results presented in this paper is a complex process. This section will discuss some potential application problem areas, result discrepancies, solutions and future work.

6.1 Complexity Vs. Visual Fidelity

It was noted that throughout the experiment, the subsurface scattering and the sample count made only a very subtle difference when combined with all of the other effects. There is no doubt that this is the most computationally expensive process in the shaders, affecting the colour of each pixel up to 200 times on a per-frame basis. Despite the relatively subtle and somewhat flawed final result, the writing process for this effect was considerably the most beneficial due to the nature and volume of the calculations. In this case the benefits of the scattering technique when compared to the adverse effect on the frame rates indicate the effect is not as necessary as others. This was made particularly evident during the perception study where no-one explicitly noticed the effect. As discussed the high resolution, dynamic cube maps were on average not necessary either. Similarly, few noticed the difference between a high and 'ultra' resolution of reflection map. In essence the most important ingredients in the rendering of eyes are some form of reflection and texture mapping. The refraction is also important at all but long range viewing distances. Consequently, reducing some of these effects or removing them completely as necessary will improve the frame throughput of any application using the eye shader suite.

6.2 Perceived Vs. Actual Detail

Some discrepancies regarding the assumptions do exist in the results, notably for the right eye in the 4th clip, and the left eye in the 6th. Both eyes in these clips maintain the 'ultra' LoD at *medium* distances. The conclusion can be drawn that neither has a higher rating of perceived realism for a potential variety of explanations. The test was not conducted in a strictly controlled environment, in part due to time constraints. This may have sub-consciously influenced the results.

Another factor is the shader results are not perfect for every layer of detail. As discussed previously and in the next section some calculations are approximated. In some cases this leads to a less satisfactory result than intended: the highest level of realism should really be photo-realistic at all distances rather than just macroscopic instances. Another significant issue relates to the limited nature of the survey. More accurate results would of course be achieved with more test subjects, and many more videos and iterations of the eye renderings.

An often overlooked issue is the nature of subconscious decision making during surveys. Friedman and Amoo [30] note that people have a remarkable tendency to ignore either the maximum or minimum option for a rating scale in a description survey such as this. As an unforeseen side-effect, this may also explain why only 6 out of 140 realism evaluations were a perfect 5, and 3 of those were from the same test subject. A solution for future work would be a much larger and more detailed survey.

6.3 Criticisms, Solutions and Future Work

The creation process of this project featured several key obstacles. The subsurface texture mapping was specifically difficult to implement. The only available resource is the paper from which the effect was first published [28]. When coding the effect, the paper features two deceptively complicated algorithms in pseudo-code which required a steep learning curve to dissect and debug.

The refraction implementation is also relatively novel in comparison to standard techniques summed up in graphics rendering texts. Because the subsurface texture mapping occurs beneath the surface of the refracted object, it was particularly difficult to realize a form which would account for this as well. This is because the subsurface mapping relies on the refracted light vector, whereas the refraction relies on rendering what is underneath the surface first. This means the current subsurface texture mapping result will only completely work without the refraction. Hence, the subsurface scattering that is achieved in the demo presents the effect from one angle only (but is witnessed at all angles). This circular dependency remains an area of future work, but with this constraint the results remain satisfactory even at close distances.

Another area of improvement might be to implement an 'unrefracted' iris photo-

graph. A photograph of an iris is naturally refracted because it is being viewed through the cornea. François et al. [28] unrefract an iris photograph in an offline pre-processing step which uses an image based technique to do so. This is then refracted again in real-time in order to apply subsurface texture mapping to the iris. For the purposes of this paper an iris photograph was unrefracted by hand in Adobe Photoshop™. The result however remains relatively satisfactory without using an automated unrefraction process.

Generating real-time caustics is another area of improvement for the application. Reliable caustics are a wide and current field of graphics study, and perhaps worth an entire paper alone. The implementation that is used is a simple approximation in comparison to the current forms of caustics described in Chapter 2.7. By using one of these techniques instead, a higher level of fidelity would be achieved, particularly at close ranges.

Finally for game purposes the eyes currently do not seamlessly switch between different levels of realism; which was not the focus of this paper. Instead when moving the camera toward the character, the transitions are somewhat noticeable at certain angles. This was not a problem during the perception study because each eye maintained a constant LoD for the duration of each video. For game-ready eyes, the correct distance based blending of the materials would satisfactorily lead to a smooth, unnoticeable switching in resolutions. LoD transitioning remains a noticeable problem even in current games, and is one facet of future work that could be undertaken for a more realistic effect.

6.4 Epilogue

This study has presented and addressed a multitude of concepts regarding the creation and sustainment of real-time eyes in virtual characters. The eyes remain one of the most important features of the human body, and only now are developers beginning to afford graphical resources to cloning eyes in as realistic a manner as possible. By exploiting the computational horse power of modern programmable GPUs in an effective manner, photo-realistic eyes may be rendered on arbitrary crowds of virtual characters. The novel approach to shading virtual eyes in this paper defines a flexible manner in which to designate varying levels of realism for the eyes. There are 7 core effects which work in tandem to produce an eye of closely photographic visual fidelity. There are 4 iterations of the each core effect which demonstrate the efficacy and practicality of LoD functionality in all aspects of real-time rendering. To date François et al. [28] have created the most in-depth and faithful virtual eye result, but with no level-of-detail consideration. Modern games such as *Beyond : Two Souls* from Quantic Dreams™, or the next generation *Luminous* engine from Square Enix™provide a glimpse of what can be expected in the near future (see Figures 6.2 and 6.1).

This paper discourses the repercussions and extensibility of such an implementation. It is hoped that real-time applications in the future sustain both photo-realism and real-time frame rates simultaneously. As the parallel processor returns of future GPUs begins to diminish, software engineers must rely on techniques such as the system presented in this paper to attain that goal.

Figure 6.1: *The Luminous Engine from Square Enix*^{TM}*promises groundbreaking visual effects.*

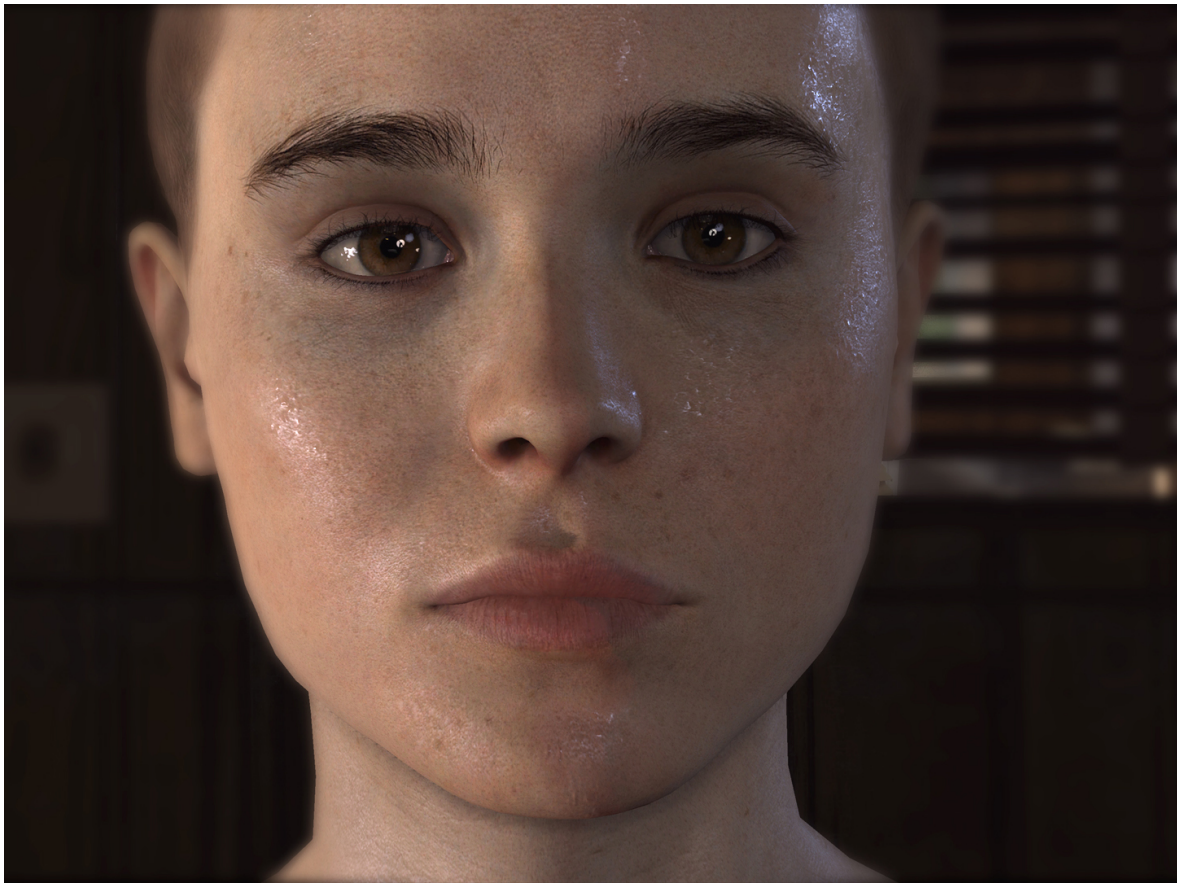


Figure 6.2: *Actress Ellen Page reimagined in upcoming cinematic thriller Beyond : Two Souls™.*

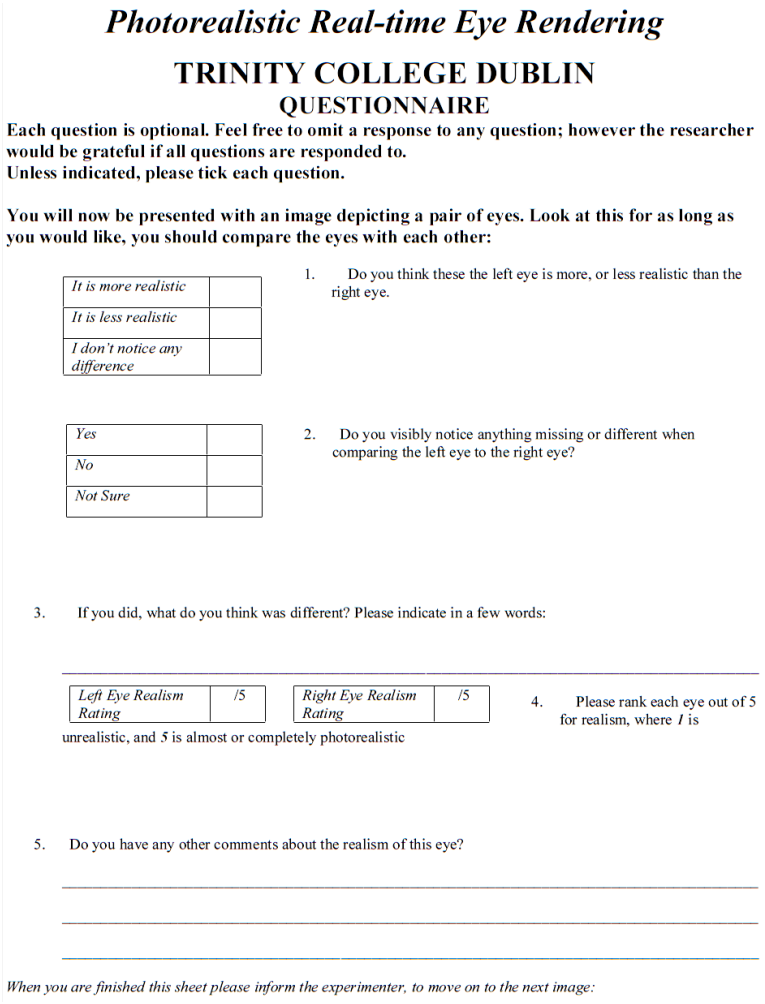
Appendix A

```
vertex_program no_specular cg
{
    source Blinn_Phong.cg
    entry_point v_shader
    profiles vs_2_0 arbvp1
    compile_arguments -SPECULAR=0
    default_params{param_named_auto WorldXf world_matrix}
}
material Blinn_Phong
{
    lod_strategy eye_distance_pixels
    technique
    {
        lod_index 0
        pass
        {
            vertex_program_ref vp{}
            fragment_program_ref fp{}

            texture_unit
            {
                texture diffuse
            }
        }
    }
    technique //Fall-back technique which uses no specular contribution
    {
        lod_index 1
        pass
        {
            vertex_program_ref no_specular_vp{}
            fragment_program_ref no_specular_fp{}
            texture_unit{ texture diffuse }
        }
    }
}
```

Figure 3: *This simple script in Ogre defines a Blinn-Phong material which contains techniques, passes, LoD functionality and pragma statements. The pragma statement: -SPECULAR indicates that a specular component will not be used in this pass.*

Appendix B



Photorealistic Real-time Eye Rendering

TRINITY COLLEGE DUBLIN

QUESTIONNAIRE

Each question is optional. Feel free to omit a response to any question; however the researcher would be grateful if all questions are responded to.
Unless indicated, please tick each question.

You will now be presented with an image depicting a pair of eyes. Look at this for as long as you would like, you should compare the eyes with each other:

It is more realistic	
It is less realistic	
I don't notice any difference	

1. Do you think these the left eye is more, or less realistic than the right eye.

Yes	
No	
Not Sure	

2. Do you visibly notice anything missing or different when comparing the left eye to the right eye?

3. If you did, what do you think was different? Please indicate in a few words:

_____

Left Eye Realism Rating	/5	*Right Eye Realism Rating*	/5

4. Please rank each eye out of 5 for realism, where *1* is unrealistic, and *5* is almost or completely photorealistic

5. Do you have any other comments about the realism of this eye?

_____

_____

_____

When you are finished this sheet please inform the experimenter, to move on to the next image:

Figure 4: *The perception study featured one sheet for each video. Subjects were also asked about their experience with computer games, their age, gender and eyesight ability.*

Bibliography

[1] Akenine-Möller, T., Haines, E., and Hoffman, N. Caustics. In *Real-Time Rendering 3rd Edition*. A. K. Peters, Ltd., 2008, pp. 399–401.

[2] Akenine-Möller, T., Haines, E., and Hoffman, N. Glossy reflection maps. In *Real-Time Rendering 3rd Edition*. A. K. Peters, Ltd., 2008, pp. 308–313.

[3] Akenine-Möller, T., Haines, E., and Hoffman, N. *Real-Time Rendering 3rd Edition*. A. K. Peters, Ltd., Natick, MA, USA, 2008.

[4] Akenine-Möller, T., Haines, E., and Hoffman, N. Subsurface scattering theory. In *Real-Time Rendering 3rd Edition*. A. K. Peters, Ltd., 2008, pp. 403–407.

[5] AMD. Gpu shader analyzer. `http://developer.amd.com/tools/shader/Pages/default.aspx`, 2012. Last Accessed 21/05/12.

[6] Andujar, C., Boo, J., Brunet, P., Fairen, M., Navazo, I., Vazquez, P., and Vinacua, A. Omni-directional relief impostors. *Computer Graphics Forum 26*, 3 (2007), 553–560.

[7] Autodesk. Rendering glass and ice. `http://www.rastertek.com/dx10tut32.html`, 2012. Last Accessed 26/08/12.

[8] Baker, T. Y. Ray tracing through non-spherical surfaces. *Proceedings of the Physical Society 55*, 5 (1943), 361.

[9] Baroody, A. Tutorial: Its all in the eyes. `http://www.3dluvr.com/rogueldr/tutorials/eye/eyes.html`, 2010. Last Accessed 21/05/12.

[10] Blinn, J. F. Models of light reflection for computer synthesized pictures. *SIGGRAPH Comput. Graph. 11*, 2 (July 1977), 192–198.

[11] Blinn, J. F., and Newell, M. E. Texture and reflection in computer generated images. *Commun. ACM 19*, 10 (Oct. 1976), 542–547.

[12] Brenton, H., Gillies, M., Ballin, D., and Chatting, D. D.: The uncanny valley: does it exist. In *In: 19th British HCI Group Annual Conference: workshop on human-animated character interaction* (2005).

[13] Buss, S. R. *3D Computer Graphics: A Mathematical Introduction with OpenGL.* Cambridge University Press, New York, NY, USA, 2003.

[14] Catmull, E. E. *A subdivision algorithm for computer display of curved surfaces.* PhD thesis, 1974. AAI7504786.

[15] Cebenoyan, C. Ch 28 : Graphics pipeline performance. In *GPU Gems: Programming Techniques, Tips and Tricks for Real-Time Graphics*, R. Fernando, Ed. Addison-Wesley, 2004, pp. 473–486.

[16] Chandrasekhar, S. *Radiative Transfer.* Courier Dover Publications, Dover, UK, 1960.

[17] Cook, R. L., and Torrance, K. E. A reflectance model for computer graphics. *SIGGRAPH Comput. Graph. 15*, 3 (Aug. 1981), 307–316.

[18] Davies, Martin. Gdc 2012: David cage shows new tech demo. `http://www.edge-online.com/news/gdc-2012-david-cage-shows-new-tech-demo`, 2012. Last Accessed 21/05/12.

[19] d'Eon, E., and Luebke, D. Advanced techniques for realistic real-time skin rendering. In *Gpu Gems 3*, H. Nguyen, Ed. Addison-Wesley Professional, 2007, pp. 403–407.

[20] Donner, C., and Jensen, H. W. Light diffusion in multi-layered translucent materials. *ACM Trans. Graph. 24*, 3 (2005), 1032–1039.

[21] Eagle, R. Iris pigmentation and pigmented lesions: An ultrastructural study. *Trans. Am. Ophthalmological Soc. 86* (1988), 581–687.

[22] Elhelw, M., Nicolaou, M., Chung, A., Yang, G.-Z., and Atkins, M. S. A gaze-based study for investigating the perception of visual realism in simulated scenes. *ACM Trans. Appl. Percept. 5*, 1 (Jan. 2008), 3:1–3:20.

[23] Ennis, C., McDonnell, R., and O'Sullivan, C. Seeing is believing: body motion dominates in multisensory conversations. *ACM Trans. Graph. 29* (July 2010), 91:1–91:9.

[24] Epic Community Forum. Creating a skybox from a cube-map. `http://forums.epicgames.com/threads/899091-Creating-a-skybox-from-cubemap`, 2012. Last Accessed 28/08/12.

[25] Ernst, M., Akenine-Möller, T., and Jensen, H. W. Interactive rendering of caustics using interpolated warped volumes. In *Proceedings of Graphics Interface 2005* (School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2005), GI '05, Canadian Human-Computer Communications Society, pp. 87–96.

[26] Ewins, J. P., Waller, M. D., White, M., and Lister, P. F. Mip-map level selection for texture mapping. *IEEE Transactions on Visualization and Computer Graphics 4*, 4 (Oct. 1998), 317–329.

[27] Fernando, R., and Kilgard, M. J. Chapter 7 : Environment mapping techniques. In *The Cg Tutorial: The Definitive Guide to Programmable Real-Time Graphics*. Addison-Wesley Longman Publishing Co., Inc., 2007, p. 296.

[28] François, G., Gautron, P., Breton, G., and Bouatouch, K. Image-based modeling of the human eye. *IEEE Transactions on Visualization and Computer Graphics 15*, 5 (Sept. 2009), 815–827.

[29] François, G., Pattanaik, S., Bouatouch, K., and Breton, G. Subsurface texture mapping. In *ACM SIGGRAPH 2006 Sketches* (New York, NY, USA, 2006), SIGGRAPH '06, ACM.

[30] Friedman, H. H., and Amoo, T. Rating the rating scales. *Journal of Marketing Management 9*, 9-3 (1999), 114–123.

[31] Gillispie, C. *Pierre Cabanis - Heinrich Von Dechen*. Dictionary of scientific biography / [American Council of Learned Societies]. Charles Coulston Gillispie [Vol. 17 ff. Frederic L. Holmes], ed. in chief. Scribner, 1971.

[32] Greene, N. Environment mapping and other applications of world projections. *IEEE Comput. Graph. Appl. 6*, 11 (Nov. 1986), 21–29.

[33] Greene, S. Ch 16 : Real-time approximations to subsurface scattering. In *GPU Gems: Programming Techniques, Tips and Tricks for Real-Time Graphics*, R. Fernando, Ed. Addison-Wesley, 2004, pp. 263–278.

[34] Hecht, E., and Zajac, A. *Optics*, 2nd ed. Addison-Wesley, 1997.

[35] Hegeman, K., Ashikhmin, M., and Premože, S. A lighting model for general participating media. In *Proceedings of the 2005 symposium on Interactive 3D graphics and games* (New York, NY, USA, 2005), I3D '05, ACM, pp. 117–124.

[36] Hertzmann, A., O'Sullivan, C., and Perlin, K. Realistic human body movement for emotional expressiveness. In *ACM SIGGRAPH 2009 Courses* (New York, NY, USA, 2009), SIGGRAPH '09, ACM, pp. 20:1–20:27.

[37] Hillaire, S. Translucent dual relief mapping. http://sebastien.hillaire.free.fr/index.php?option=com_content&view=article&id=58&Itemid=69, 2011. Last Accessed 21/05/12.

[38] Huth, G. A modern explanation for light interaction with the retina of the eye based on nanostructural geometry: Rethinking the vision process. http://www.ghuth.com/, 2010. Last Accessed 21/05/12.

[39] Imesch, P., Bindley, C., Khademian, Z., Ladd, B., Gangnon, R., Albert, D., and Wallow, I. Melanocytes and iris color. electron microscopic findings. *Archives of Opthalmology 114*, 4 (1996), 443–447.

[40] Irrlicht. Irrlicht home page. http://irrlicht.sourceforge.net/, 2012. Last Accessed 26/08/12.

[41] Jensen, H. W., Marschner, S. R., Levoy, M., and Hanrahan, P. A practical model for subsurface light transport. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), SIGGRAPH '01, ACM, pp. 511–518.

[42] Jensen, H. W., Marschner, S. R., Levoy, M., and Hanrahan, P. A practical model for subsurface light transport. In *SIGGRAPH* (2001), pp. 511–518.

[43] Jonathan Dummer. Cone step mapping: An iterative ray-heightfield intersection algorithm. http://www.lonesock.net/files/ConeStepMapping.pdf, 2006. Last Accessed 21/05/12.

[44] Khronos Group. Opengl home page. http://www.opengl.org/, 2012. Last Accessed 26/08/12.

[45] Kniss, J., Premoze, S., Hansen, C., Shirley, P., and McPherson, A. A model for volume lighting and modeling. *IEEE Transactions on Visualization and Computer Graphics 9*, 2 (Apr. 2003), 150–162.

[46] Koci, Roberto. World, view and projection matrix unveiled. http://robertokoci.com/world-view-projection-matrix-unveiled/, 2012. Last Accessed 26/08/12.

[47] Lefohn, A., Budge, B., Shirley, P., Caruso, R., and Reinhard, E. An ocularist's approach to human iris synthesis. *IEEE Comput. Graph. Appl. 23*, 6 (Nov. 2003), 70–75.

[48] Lindstrom, P., Koller, D., Ribarsky, W., Hodges, L. F., Faust, N., and Turner, G. A. Real-time, continuous level of detail rendering of height fields. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1996), SIGGRAPH '96, ACM, pp. 109–118.

[49] Luebke, D., Watson, B., Cohen, J. D., Reddy, M., and Varshney, A. 6.5 nongeometric level of detail. In *Level of Detail for 3D Graphics*. Elsevier Science Inc., 2002, p. 171.

[50] Mann, M. The nervous system in action - chapter 7: Vision. `http://michaeldmann.net/mann7.html`, 1997. Last Accessed 21/05/12.

[51] Masters, B. Three-dimensional microscopic tomographic imagings of the cataract in a human lens in vivo. *Opt. Express 3*, 9 (Oct 1998), 332–338.

[52] Matthijs De Smedt. Augmenting the pc graphics of deus ex: Human revolution using directx 11 technology. `http://www.gdcvault.com/play/1015508/Advanced-Visual-Effects-with-DirectX`, 2012. Last Accessed 21/05/12.

[53] McDonnell, R., and Breidt, M. Face reality: investigating the uncanny valley for virtual faces. In *ACM SIGGRAPH ASIA 2010 Sketches* (New York, NY, USA, 2010), SA '10, ACM, pp. 41:1–41:2.

[54] McMillan, Jr., L. *An image-based approach to three-dimensional computer graphics*. PhD thesis, Chapel Hill, NC, USA, 1997. UMI Order No. GAX97-30561.

[55] Mertens, T., Kautz, J., Bekaert, P., Reeth, F. V., Seidel, H.-P., Van, F., peter Seidel, R. H., and Centrum, L. U. Efficient rendering of local subsurface scattering. In *In Proceedings of the 11th Pacific Conference on Computer Graphics and Applications* (2003), p. 51.

[56] Michel, B. S. H. S. General-purpose gpu computing: practice and experience. In *Proceedings of the 2006 ACM/IEEE conference on Supercomputing* (New York, NY, USA, 2006), SC '06, ACM.

[57] Microsoft. Msdn xbox live indie games. `http://xbox.create.msdn.com/en-us`, 2012. Last Accessed 26/08/12.

[58] Mori, M. Bukimi no tani [The uncanny valley]. *Energy 7*, 4 (1970), 33–35.

[59] MSDN. World transform (direct3d 9). `http://msdn.microsoft.com/en-us/library/windows/desktop/bb206365(v=vs.85).aspx`, 2012. Last Accessed 26/08/12.

[60] Nayar, S., and Oren, M. Generalization of the Lambertian Model and Implications for Machine Vision. *International Journal on Computer Vision 14*, 3 (Apr 1995), 227–251.

[61] Nedelman, Derek. Ogremax exporter. `http://www.ogremax.com/`, 2012. Last Accessed 26/08/12.

[62] Nijdam, N. A. Facial Rendering. Master's thesis, Universiteit Twente, 2007.

[63] Nishita, T. Light scattering models for the realistic rendering of natural scenes. In *Rendering Techniques* (1998), pp. 1–10.

[64] NVidia Corporation. Gpu programming guide geforce 8 and 9 series. `http://developer.download.nvidia.com/GPU_Programming_Guide/GPU_Programming_Guide_G80.pdf`, 2008. Last Accessed 21/05/12.

[65] NVidia Corporation. Perfhud 6. `(http://developer.nvidia.com/nvidia-perfhud`, 2012. Last Accessed 21/05/12.

[66] NVidia Developer Zone. Fx composer 2.5. `http://developer.nvidia.com/content/fx-composer`, 2012. Last Accessed 26/08/12.

[67] Oberg, E. Procedural eyes. `http://www.inear.se/2010/08/procedural-eyes/`, 2010. Last Accessed 21/05/12.

[68] Ogre 3D. Ogre 3d home page. `http://www.ogre3d.org/`, 2012. Last Accessed 26/08/12.

[69] Olano, M., Kuehne, B., and Simmons, M. Automatic shader level of detail. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware* (Aire-la-Ville, Switzerland, Switzerland, 2003), HWWS '03, Eurographics Association, pp. 7–14.

[70] Oliveira, M. *Relief Texture Mapping*. PhD thesis, Chapel Hill, NC, USA, 2000. UNC Computer Science Technichal Report.

[71] Oliveira, M. M., Bishop, G., and McAllister, D. Relief texture mapping. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2000), SIGGRAPH '00, ACM Press/Addison-Wesley Publishing Co., pp. 359–368.

[72] Oliveira, M. M., and Brauwers, M. Real-time refraction through deformable objects. In *In Proceedings of the 2007 Symposium on Interactive 3D graphics and games (2007* (2007), pp. 89–96.

[73] O'Sullivan, C., and Dobbyn, S. Populating Virtual Environments with Crowds: Level of Detail for Real-Time Crowds. Eurographics Tutorial, 2006.

[74] Pharr, M., and Fernando, R. *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation (Gpu Gems).* Addison-Wesley Professional, 2005.

[75] Phong, B. T. Illumination for computer generated pictures. *Commun. ACM 18*, 6 (June 1975), 311–317.

[76] Policarpo, F., and Oliveira, M. M. Relief mapping of non-height-field surface details. In *Proceedings of the 2006 symposium on Interactive 3D graphics and games* (New York, NY, USA, 2006), I3D '06, ACM, pp. 55–62.

[77] Policarpo, F., and Oliveira, M. M. Relaxed cone stepping for relief mapping. In *Gpu Gems 3*, H. Nguyen, Ed. Addison-Wesley Professional, 2007, pp. 123–126.

[78] Policarpo, F., Oliveira, M. M., and Comba, J. a. L. D. Real-time relief mapping on arbitrary polygonal surfaces. In *Proceedings of the 2005 symposium on Interactive 3D graphics and games* (New York, NY, USA, 2005), I3D '05, ACM, pp. 155–162.

[79] Preisz, E., and Garney, B. *Video Game Optimization*, 1st ed. Course Technology Press, Boston, MA, United States, 2010.

[80] Pstchoart. Zbrush render bpr + fibermesh. `http://www.zbrushcentral.com/showthread.php?166136-Zbrush-Render-BPR-Fibermesh`, 2012. Last Accessed 26/08/12.

[81] RasterTek. 3ds max tutorial files. `http://usa.autodesk.com/adsk/servlet/item?siteID=123112&id=16759124&linkID=9241175`, 2012. Last Accessed 26/08/12.

[82] Risser, E. *True Impostors.* PhD thesis, 2006. Technical report CS-TR-07-01.

[83] Risser, E. True impostors. In *Gpu Gems 3*, H. Nguyen, Ed. Addison-Wesley Professional, 2007, pp. 481–490.

[84] Sagar, M. A., Bullivant, D., Mallinson, G. D., and Hunter, P. J. A virtual environment and model of the eye for surgical simulation. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1994), SIGGRAPH '94, ACM, pp. 205–212.

[85] Saude, T. *Ocular Anatomy and Physiology.* Wiley-Blackwell, 1993.

[86] Shrinker. Blue eyeball shader. http://www.creativecrash.com/maya/downloads/shaders/c/blue-eyeball, 2008. Last Accessed 21/05/12.

[87] Sousa, T. Ch 19 : Generic refraction simulation. In *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation (Gpu Gems)*, R. Fernando, Ed. Addison-Wesley Professional, 2005.

[88] Spaulding, T. . Entropia, tyler's graphics blog. http://www.arguingwithmyself.com/gallery.html, 2012. Last Accessed 26/08/12.

[89] Surman, D. Gaming, uncanny realism and technical demonstration. http://www.swanquake.com/usermanual/DavidSurman/index.htm, 2006. Last Accessed 21/05/12.

[90] Taha, Abderrahman. K3dsurf. http://k3dsurf.sourceforge.net/, 2012. Last Accessed 26/08/12.

[91] Talden, W. Talden's graphics blog. http://www.talden.com/Pages/home_TEXT.htm, 2012. Last Accessed 26/08/12.

[92] Tatarchuk, N. and Barczak, J. and Bilodeau, B. Programming for real-time tessellation on gpu. http://developer.amd.com/gpu_assets/Real-Time_Tessellation_on_GPU.pdf, 2009. Last Accessed 21/05/12.

[93] The Sims Studio. The sims 3. http://www.thesims3.com/, 2012. Last Accessed 26/08/12.

[94] Tuchin, V. V., Maksimova, I. L., Mishin, A. A., and Mavlutov, A. K. Scleral tissue light scattering and matter diffusion. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series* (June 1998), P. O. Rol, K. M. Joos, and F. Manns, Eds., vol. 3246 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, pp. 249–259.

[95] TurboSquid. Turbosquid home page. http://www.turbosquid.com/, 2012. Last Accessed 26/08/12.

[96] UBC Mathematics. The law of refraction. http://www.math.ubc.ca/~cass/courses/m309-01a/chu/Fundamentals/snell.htm, 2012. Last Accessed 26/08/12.

[97] Unity Technologies. Unity home page. http://unity3d.com/, 2012. Last Accessed 26/08/12.

[98] Wang, H., Lin, S., Liu, X., and Kang, S. B. Separating reflections in human iris images for illumination estimation. *Computer Vision, IEEE International Conference on 2* (2005), 1691–1698.

[99] Wang, R., Tran, J., and Luebke, D. All-frequency interactive relighting of translucent objects with single and multiple scattering. *ACM Trans. Graph. 24*, 3 (July 2005), 1202–1207.

[100] Wiley, Abe. Authoring for real-time tessellation and displacement mapping. `http://developer.amd.com/gpu_assets/WileyAuthoringforTessellation.pdf`, 2009. Last Accessed 21/05/12.

[101] Wilhelmsen, Petri. Xna shader programming. `http://digitalerr0r.wordpress.com/2009/05/03/xna-shader-programming-tutorial-16-refraction/`, 2012. Last Accessed 26/08/12.

[102] W.Y. Lam, M., and V.G. Baranoski, G. A predictive light transport model for the human iris. *Computer Graphics Forum 25*, 3 (2006), 359–368.

[103] Wyman, C. Hierarchical caustic maps. In *Proceedings of the 2008 symposium on Interactive 3D graphics and games* (New York, NY, USA, 2008), I3D '08, ACM, pp. 163–171.

[104] Xavier, B. Faster refraction formula, and transmission color filtering. `http://tog.acm.org/resources/RTNews/html/rtnv10n1.html#art3`, 1997. Last Accessed 21/05/12.

[105] Zinkova, M. . Caustic network and barracuda. `http://epod.usra.edu/blog/2009/07/caustic-network-and-barracuda.html`, 2012. Last Accessed 28/08/12.

[106] Zuo, J., and Schmid, N. A. A model based, anatomy based method for synthesizing iris images. In *In Springer LNCS 3832: Int. Conf. on Biometrics* (2006), pp. 486–492.