THE UNIVERSITY OF HUDDERSFIELD

Photorealism versus Non-Photorealism: Art styles in computer games and the default bias.

Master of Research (MRes) Thesis

Nathan Jarvis - U0859020010

18/09/2013

Supervisor: Daryl Marples
Co-Supervisor: Duke Gledhill

1.0.0 – Contents.

2.0.0 – Abstract.

This thesis looks into the prevalence of photorealism in computer games design in comparison to non-photorealism. The aim is to develop an understanding of the reason for this prevalence, and with this knowledge look into ways that non-photorealistic rendering (NPR) can be expanded or improved upon to bring balance to the two styles. This is done primarily through research utilising both journalistic and academic sources to seek existing opinions and theories on the subject, with subsequent experiments being used to provide evidence of whether these theories are true or not.

Two potential reasons were considered for the prevalence of photorealism, the first being that photorealism was simply the more popular style. Existing theories showed the possibility that photorealism is favoured amongst consumers for providing a more immersive and mature experience, with non-photorealism potentially having negative associations with children's media by way of their cartoon-like imagery. A subsequent survey into the opinions of a group of consumers proved this to be incorrect; neither style is more popular than the other, both are in fact favoured equally.

With this proven, a second potential reason was explored; that photorealism was the easier style to produce. Existing research shows that photorealism is a natural fit for 3D rendering due to the both of them being technical art forms. Non-photorealism on the other hand is a form that requires much in the way of personal interpretation, a facet that is difficult for a computer to replicate without a significant amount of input from the user. This phenomenon can be described as the theory of the "default bias"; that 3D software (and by extension, game engines) have a default leaning towards photorealism to the detriment of non-photorealism. Subsequent experiments both proved the existence of the "default bias" within one of the most widely used game engines in the industry (the Unreal Development Kit), while at same time developing and documenting workarounds for the specific cases encountered.

The results of this research shows that there are flaws in the way some game creation tools are designed that have the potential to inhibit creativity by making any deviation from photorealism more difficult and time consuming than need be. These are flaws that need to be acknowledged. Additionally, the research is intended to be used not just by other researchers but also by those in the UDK community, the latter of which can use the information presented in chapters 7.0.0 through 9.0.0 to help produce their own non-photorealistic work.

2.1.0 – Literature Review.

Existing research into the prevalence of photorealism is limited. There are however many existing journalistic articles discussing the trend of photorealism, such as Jeremy Price's opinion piece "Is Photorealism In Games The Right Direction" which discusses Price's feelings on whether photorealism can be considered a true improvement (Price, 2006). While the article does contain a significant amount of useful information on the evolution of computer game art from the perspective of someone who has been an industry veteran for six years (which helps to further understand how the attitudes towards and methods used in games design have changed over time), it doesn't provide any solid answer to the prevalence of photorealism beyond what is essentially 'because they can', which in itself is too vague and simplistic a response.

There also exists the paper "How Realistic is Realism? Considerations on the Aesthetics of Computer Games" by Richard Wages, Stefan M. Grünvogel, and Benno Grützmacher (Wages, et al., 2004), an academic paper produced for the 3rd International Conference on Entertainment Computing (ICEC). While the research does successfully deconstruct the ways photorealism might impede the enjoyment of a game, there are unfortunately some flaws in the research that brings both its usability and integrity into question. It is a short paper with a very limited scope; its primary purpose is the aforementioned deconstruction, and little more. Similarly to Price's (2006) opinion piece it doesn't elaborate on the possibilities of why photorealism is so prevalent, instead supplying only a "straightforward" (Wages, et al., 2004, p.1) reason before continuing to deconstruct the issues of photorealism. Furthermore, when citing examples of non-photorealistic games, specifically *Donkey Kong*, *Super Mario Bros.*, and *The Legend of Zelda* (Nintendo), Wages, et al. fails to distinguish between the different instalments of each franchise. This becomes especially problematic when they refer to the special effects used in one of the *Zelda* games. Wages, et al. briefly describes how enemies, when defeated, will vanish into clouds with a "simplified flowery and spiral look" (2006, p.8). This effect is specific to *The Legend of Zelda: Wind Waker* (for the Gamecube), however, they do not at any point refer to the game by its full name and instead simply refer to it as *"Zelda*" (2006, p.8). There have been more than fifteen instalments of *The Legend of Zelda* spanning multiple consoles; someone who is not familiar to the franchise would not be able to tell which game in the series is being referenced. When discussing media in an academic manner, it is important to ensure that there is no room for misinterpretation or confusion.

Upon expanding the breadth of research into simply 'computer game graphics', the research available becomes much more plentiful. Over the last two decades there has been extensive research into the subject of computer game graphics, demonstrating new ways to achieve particular effects through shaders, textures, modelling, animation, etc. This applies both to photorealism and non-photorealism. In the case of the latter, there is the International Symposium on Non-photorealistic Animation and Rendering, an annual conference dedicated to furthering the research into non-photorealism in CGI. Each year the proceedings are published online (both as a collective and in some cases independently), demonstrating projects such as "Real-Time Watercolor Illustrations of Plants Using a Blurred Depth Test" by Thomas Luft and Oliver Deussen (Luft, et al.,2006).

This research, however, is purely technical; while it may prove useful during the latter stages of the thesis (the practical experiments in UDK), it doesn't serve to provide answers to the questions at hand (aside from indirectly proving that a lack of informative resources is *not* the reason for the lack of non-photorealism). While a technical perspective on computer game art is available in abundance, a cultural perspective on computer game art is much more limited. Any pre-existing academic research into computer gaming culture has a much stronger focus towards social issues as opposed to artistic issues, including the use of computer games as educational material (Malone, 1980), potential links between violent computer games and real-world violence (Anderson and Dill, 2000), and the representation of particular communities (Cassell, 1998). While such subjects are important, they are unrelated to the topic at hand.

To summarise, existing research into this specific topic (and the culture of computer game art in general) is incredibly limited. Any existing work that does pertain to the topic tends to come in the form of journalistic articles (in particular opinion pieces and interviews). While these opinions are

important and can be used as points of discussion, they cannot form the research alone. Instead, these opinions will need to be further explored by applying them to examples in the real world, and then further compared and contrasted with related research from other similar subjects (such as movies or marketing). Furthermore, it is necessary to make up for the absence of existing statistical data by producing new original data through an appropriate survey.

2.2.0 – Summary of Changes (September 2013).

Here is a brief summary of changes that have been made to this thesis since the initial hand-in (January 2013):

- The referencing system used has been changed from the Harvard Referencing System to the APA 6th system, in accordance with the University of Huddersfield's revised referencing standards.
- Additional sources have been included to help clarify existing information and evidence.
- One of the examples shown in section 4.2.3 (The apparent anime stigma) has been replaced with a much more appropriate example.
- Various sections of the thesis have been rewritten in order to bring clarity to both my intentions as well as the process used.
    - Previously, the connection between many of the chapters wasn't properly established, leading the thesis to feel abrupt and disjointed. The connection between these chapters has now been more clearly established, and the process of the investigation should now be much easier to follow.
    - Previously, the experiments featured did not contain clearly sectioned hypotheses or conclusions, lending confusion to their purpose. These experiments have now been reformatted and rewritten to make their purpose much clearer.
    - Numerous parts of the thesis that were found to be confusing have been simplified, with only the most necessary information on display. Furthermore, sections that I found to be poorly written have since been redone.
- The final conclusion has been restructured so that it is easier to read. Furthermore, an additional section has been added acknowledging notable examples of NPR games that have been announced and/or released since the initial hand-in.
- The title of the thesis has been changed to better reflect the content and focus.
- The model used in Experiment #4 (section 9.0.0) has been remade, as I wasn't pleased with the previous model.

2.2.1 – The use of digital publications.

One particular critique however is in the way predominantly digital publications from the internet have been referenced in this research. In order to fully understand the justification behind this, it is important to categorise the available material into the categories established by Michael Nentwich in his book *Cyberscience* (Nentwich, 2003):

- P-publishing: This refers to any work/journal/monograph that has been published through traditional printing methods (hence the prefixed P).
- E-publishing: This refers to any work/journal/monograph that has been electronically published, usually over the Internet.
- Parallel-publishing: This refers to any work/journal/monograph that is available in both electronic and digital formats, whether released simultaneously or through the digital archiving of older works.
- Pure E-publishing: This refers specifically to any work/journal/monograph that is ONLY available digitally, and is NOT available through traditional publishing.

The majority of the resources that have been used during this research can be categorised as either Parallel-works, or pure E-works. For example, the works of Messaris (1996), Mori (1970), and Brennan (1982) are Parallel-works. Although all of these were originally printed in a physical format, they have since been digitally archived and are now available over the internet either through e-book services (such as Google Books), or through academic databases.

With these categories established, I can now further explain my use of predominantly digital resources.

2.2.2 – Parallel-publishing: Why favour the electronic sources?

As established by Nentwich (2003), the internet has had a huge effect on academic research, both in the way that it is collected and in the way that it is published.

One of the ways that the internet has affected research methods is by making it easier for the researcher to find information that is relevant to their studies. This is due to two factors, the first being that anything published on the internet, either research or information, is available on a global scale. This provides researchers with a potentially much wider breadth of available information than what would be available in pure physical form. The sheer amount of information available could potentially prove to be overwhelming, leading to a case of information overload (Nentwich, 2003, p.212), hence the importance of being able to organise and sort through this information to find only the relevant work. This is the second factor; the improvement of search engine technology. With services like Google Scholar, it is now much easier to sort through this vast amount of information to find only the documents relevant to the research quickly and efficiently. (Nentwich, 2003, p.325) As Nentwich notes, “research tasks that used to take days or weeks may now be a matter of minutes” (2003, p.210). This ease of access to information has the additional effect of ensuring that research isn’t unnecessarily duplicated. (Nentwich, 2003, p.215)

The internet has also had a large effect on the way research is published. Traditionally, there would often be a significant time lag between the completion of the research, and the publication of said research. For example, typically a Mathematics journal will remain unpublished after completion for two to three years. (Nentwich, 2003, p.323) This is due to the fact that traditional paper journals are not published separately, but as a part of an annual volume. Such volumes had a limit on how many articles could be featured, leaving unpublished articles to wait each year until there is space available. (Nentwich, 2003, p.324) This time lag can prove to be problematic to the publication in

question, as it is possible that the publication "is hopelessly outdated when it finally reaches its audience." (Nentwich, 2003, p.323)

Digital publishing solves this by allowing for new articles to be published singularly online as soon as they are considered ready for publication, allowing for the publication to reach its audience whilst it is still relevant. (Nentwich, 2003, p.324) This process is made even faster by the fact that, unlike paper publishing, there is no time needed for printing or shipping as people can download the digital file (e.g. .PDF) for offline viewing. (Nentwich, 2003, p.324) Furthermore, the methods of publication are more varied; while many researchers may choose to release their works through an online publisher such as the ACM Digital Library, others can and may choose to instead self-publish the work on their personal website.

This, however, only explains how the internet has made it easier to find such resources. Many of the resources used in this thesis can be categorised as parallel-works; works that are available in both printed and electronic format. Such resources include books, journals, and conference/symposium proceedings. In many of these cases, the electronic format has been chosen over the printed format. There are various reasons for this:

1. In some cases, the work is either no longer available or difficult to obtain in a printed form. Many of the books referenced in this work have limited availability in the printed format. For example, the academic journal *Visual Imagery: Applications to Advertising* (Rossiter, 1982) is no longer available to purchase in printed format, and only a digitally archived version of Rossiter's work is available for use. Even *Cyberscience* (Nentwich, 2003), the very book being referenced here, was unavailable to purchase from any of the major book stores at the time of this writing. Fortunately, the Austrian University of Sciences (the original publishers of *Cyberscience*) have the book available on their official website as a digital e-book.

2. Digital publications are easier to organise and read through. Digital versions of such publications are capable of taking advantage of digital tools designed to aid researchers, including the ability to automatically find key-words (Nentwich, 2003, pp.325-326) as well as highlight useful passages for later use. In many cases, citations and chapter references may even be presented as URLs, allowing the reader to easily find cited sources through the internet. Not only is this one of the features of Nentwich's concept of hypertext (2003, p.266), but the e-book version of *Cyberscience* even utilises this feature for itself to provide links to both referenced works and separate chapters.

3. Immediacy. By utilising officially sanctioned digital versions of these publications, the information within can be accessed instantaneously, as opposed to waiting for a printed version to become available. Should a printed version of the work be available, additional time must be spent waiting for the work to be delivered. (Nentwich, 2003, p.326) As noted by Nentwich, with the advent of digital publishing the accessibility of academic publications "has been greatly enhanced". (2003, p.326)

It is for these reasons that the electronic versions have been used in place of the printed versions. This does not affect the validity of the work within them in any way. Regardless of whether the electronic or printed version of (for example) *Cyberscience* (Nentwich, 2003) is being used, it is still the same work. The only difference is the format used.

2.2.3 – Pure E-publishing: Can these sources be trusted?

There are, however, many sources used in this project that have no printed counterparts, and as such are known as pure E-works. The majority of these sources are journalistic; news articles concerning video games and the industry, opinion pieces, reviews, and interviews. This brings with it one primary concern amongst researchers; the integrity of these works.

The internet is a very wide and varied place populated with the many different views and opinions of its users. This can be considered both a pro and a con. The common assumption is as follows; as researchers and journalists on the internet do not need to pass their work through a publisher, they can theoretically publish anything they wish, including work that is fraudulent, plagiarised, or low in quality. (Nentwich, 2003, pp.384-385, 387)

However, as Nentwich (2003) discovered, this isn't necessarily so:

- The success of someone's work, either as a researcher or a journalist, relies heavily on their reputation. Intentionally posting low-quality/incorrect information has the potential to hurt or ruin their reputation, which in turn can hurt their future work. (Nentwich, 2003, p.387). Furthermore, the same search engine technology mentioned in section 2.2.2 can be easily used to contrast and compare documents and information, making the detection of plagiarised or fraudulent information much easier. This higher risk of being exposed "as a dishonest, non-professional researcher" (Nentwich, 2003, pp.437-438) alone can be considered a deterrent from these actions. This is typically referred to as a "self policing" effect. (Harnad, 1998 cited in Nentwich, 2003, p.387)

- Should a digital publication accidentally contain erroneous information, such mistakes can be easily rectified and the fixed article re-uploaded without issue. This is not the case for mass-printed works. Such editing can also be used to regularly update particular data or research to keep it from becoming outdated, allowing the publication to remain relevant for longer. Such updates can even be publically documented, allowing for complete transparency as well as access to older versions. (Nentwich, 2003, p.457) This also applies to online journalism; for example, a news article concerning a widely used service being hacked may be subsequently updated with important information (such as what customer data was compromised) as soon as it becomes available.

Despite this reasoning, many still consider E-media to be untrustworthy; compared to traditionally printed works there is a distinct lack of prestige. (Nentwich, 2003, p.353) The printed text has an air of authority due to its significant and long history, something which the relatively young digital format lacks. As explained by Guedon, the printed text has been associated with "authority and power" since the very beginning due to its access being strictly controlled, limiting its availability only to the privileged few. This is a significant contrast to the internet, which in its comparatively short history has been considered an open and unregulated format and as such does not command the same authority. (1994 cited by Nentwich, 2003, p.354)

While it is important to both remember and celebrate the history of the printed text, prestige (or lack thereof) alone cannot be used as adequate judgement for a piece of work. While there are many examples on the internet of claims and opinions that are unprofessional and poorly researched, there also exist examples of pure E-works on the internet that are trustworthy and

professional. A good example of such a source is Gamasutra, a website operated by UBM TechWeb. Gamasutra originally began as the online version of the printed magazine Game Developer, also developed by UBM TechWeb. While most conventional gaming magazines focus primarily on computer game reviews and previews, Game Developer instead focused on games from an industry perspective, being primarily marketed towards working and aspiring game creators. Both Gamasutra and Game Developer ran concurrently from 1997 until July of 2013, when UBM TechWeb officially ended Game Developer. Gamasutra has since continued as both an archive of Game Developer articles and as a source of its own pure E-articles.

Published works, whether they be journals, articles, or monographs, should be judged not on the method of which they are distributed (printed or digital), but on their content. With this in mind, every precaution has been taken to ensure that the works referenced in this research are works that are both relevant and trustworthy, regardless of the format they use.

3.0.0 – Introduction.

3.1.0 – The importance of computer game visuals.

In order to understand the importance of the research, it is necessary to understand the importance of computer game visuals in general. Within the context of games design, there are various components that are important in the production of a successful game. Visuals are but one of these components, the others consisting of elements such as story, audio, and gameplay. Which of these components is of higher priority differs depending on the person, both consumer and developer alike.

For example, gaming journalist LeClair (2012) list his preferred key components as immersion, music, addictiveness, and visuals. Although he acknowledges that successful implementation of these components "does not guarantee that a game will be good", he does place a significant amount of importance on these components, noting that should a game fail to implement them in a satisfactory way, "you can almost guarantee that a game will not be nearly as great as it could have been". (LeClair, 2012)

Antoniades (chief designer for Ninja Theory's *Enslaved*) on the other hand, places far more importance on story. In an interview with Gamasutra he emphasised how important the game's story is to him, stating that he believes it is "the most important part of the game experience". Acknowledging the importance of other elements such as gameplay, sounds, and cinematics, he argues that "all of those things are a part of the story". (Antoniades, 2010)

Both LeClair (2012) and Antoniades (2010) place differing levels of importance on different game design components; Antoniades places a large and detailed storyline as the most important element, while LeClair prefers a more visceral experience. Furthermore, the viewer comments below each article showcase a wide variety of readers sharing their own thoughts and opinions on the subject, with some agreeing with the opinions shared and others disagreeing while sharing their own. Both LeClair and Antoniades, as well as the viewers of each article, each have their own opinion as to which elements are most important. With this in mind, it is easy to assume that the importance of visuals is highly subjective.

Despite this however, there is still a large amount of importance placed on visuals by the gaming community as a whole. One of the ways that can be seen is through the technological demos used by those working within the industry. Technological demos (henceforth referred to as tech demos) are demonstrations created with the intent of showcasing the possible applications of current technology in an enclosed and experimental manner. These tech demos are predominantly used as a form of marketing, showing the potential of either the hardware or engine the company in question has produced. One of the more well known examples of this is NVidia's *Dawn* demo, used to advertise their line of graphics cards at the time. The demo shows a realistically rendered pixie walking along a branch in a very sparse and low detail environment. It doesn't exist to showcase the music, nor does this tech demo have a story. It also has very limited interaction, only allowing the viewer to move the camera and choose which animation is playing. It exists solely to showcase the graphical capabilities of real-time 3D at the time, showing the capabilities of the graphics card.

This of course makes sense within the context of computer gaming hardware; a company that produces graphics cards, hardware dedicated to rendering realtime 3D, would want to show the capabilities of said cards, and a tech demo concentrating on graphics is the best way to do this. It wouldn't make sense for Nvidia to incorporate storylines or advanced interaction into their demos as these components do not affect the capabilities of the graphics card.

A game engine on the other hand is a multi-use product with various features that are important to game development, including physics, animation tools, artificial intelligence, and coding tools. In spite of this, tech demos for game engines will still largely concentrate on showing the visual impact these engines are capable of. Tech demos such as Epic's *Samaritan* demo and Square-Enix's *Agni's Philosophy* demo choose to specifically showcase the visuals their technology is capable of achieving.

For example, the *Samaritan* demo starts with a slow pan of a futuristic city. Much of the background is out of focus, showcasing the new bokeh effect that is possible. The streets are slick with rain, showcasing the Unreal engine's new image-based reflections. After the appearance of the official "Powered By Unreal Technology" logo, the viewer is greeted to the main character, the Samaritan. He is a generic tough looking male protagonist, wearing a long leather coat powered by NVidia PhysX cloth physics. As he brings the welding torch to his face to light his cigarette, the viewer gets a clear look at how realistically the light diffuses through his skin with UDK's new subsurface scatter shader. Further close ups of his face show the realistic detailing of both his skin and facial hair. Hearing sirens, he chooses to investigate. The Samaritan can see a group of four futuristic police officers, all of them wearing leather coats powered by NVidia PhysX, senselessly beating up an innocent old man. Upon seeing this, the Samaritan through another closeup, dynamically transforms into a beast with armour plated skin right before the viewer's eyes, an effect possible only with Tessellation (a feature now available in the Unreal engine as part of its implementation of DirectX11). A fight scene ensues, showcasing the engine's ability to dynamically animate the aforementioned leather coats even in more complex scenes. Once the police are dead, a large robot appears, and the demo ends on a cliffhanger. While this three minute demo has what can be considered a basic plot, the plot *Samaritan* is there only to justify the use of specific designs and special effects that best showcase the visual capabilities of the engine.

This emphasis on visuals is a trait shared both by the industry as well as the fan community. This can be seen in the way members of the fan community have compared the graphical quality of games

released for both the Playstation3 and Xbox 360 as a way to see which one is the most powerful. For example, searching for "Xbox 360 vs PS3 comparison" on YouTube will bring up many videos created by journalists and fans alike comparing each version of numerous popular games, including; *Assassin's Creed*, *Max Payne 3*, *GTA IV*, *Dishonored*, *Battlefield 3*, *NeverDead*, *Call of Duty 3*, *Elder Scrolls V: Skyrim*, *LA Noire*, *Resident Evil 6*, *Final Fantasy XIII*. While these are all different games that provide different experiences, they all go through the same comparative process to decide which of the two consoles provides the most visually superior experience.

While the importance of visuals may vary on an individual basis, there is at the least a large amount of emphasis placed on the visuals when viewing both the industry and the community as a collective.

One potential reason for this may be due to the visuals being the most obvious of aspects. Much like movies, video games are a visual medium. Not only do video games require visuals to easily communicate the story to the viewer (just like their movie counterparts), but video games also require the use of visual cues and feedback so that the game can be played. Whether this is in the form of a heads up display or in the visual behaviour of the opponent(s), games require clear visuals to be enjoyed to their full potential.

Once it has been established that games are a visual medium similar to movies, it can then be extrapolated that both rely heavily on their use of visuals in marketing. A movie trailer relies on the use of visuals to convey various selling points to the audience in a way that is clear and concise; the premise, the featured actors, the genre, etc. A video game trailer is very similar in this regard, relying on the use of visuals to convey its own selling points; the premise, the genre, the gameplay, etc. Gameplay in particular is difficult to portray through words, and as such requires visuals to get the point across (e.g the difference between describing the Kinect controller and seeing it in use). With this in mind it can be deduced that visuals are of high importance in the marketing of video games.

This isn't exclusive to video game advertising however; visuals have long been an important element in the world of advertising. John R. Rossiter of Columbia University discusses the importance and power of visual imagery within advertising in his work *Visual Imagery: Applications to Advertising*. In this, Rossiter (1982, G-1) explains that "pictures have a well-known superiority over words when it comes to learning (important for brand awareness and brand beliefs)", and that this is due in part to the longevity of long-term visual memory in comparison to long-term verbal memory. "Long –term visual memory[…]appears to have virtually unlimited capacity, deteriorates very slowly, [if] at all, and shows no primacy of recency effects". (Avons and Phillips, 1980 cited in Rossiter, 1982, G-1)

The works of Grass and Wallace (1974) and Media Book Inc. (1979 c), as cited in Rossiter (1982, G-1), have both proven the effectiveness of TV advertising (exclusively visual) to other forms of advertising such as print ads and radio.

Furthermore, a visual stimulus in advertising has a much higher chance of achieving an emotional response than verbal advertising. This phenomenon is labelled by Messaris (1996, p.3) as the image's *iconicity* property. He notes the fact that such images are capable of "[recreating] the kinds of visual information that our eyes and brains [use] when we look at the real world" (Messaris, 1996, p.3), and that an individual's visual perception of the real world "is intimately connected with emotion" (Messaris, 1996, p.4), and that theoretically visual imagery can be used to exploit these "response tendencies" (Messaris, 1996, p.4) through its ability to recreate this visual information.

One example of this connection between visual cue and emotional response comes from advertising featuring someone looking directly at the viewer, a tactic commonly used by TV spokespeople and models in magazines. This technique relies on the human tendency to look back at someone who is looking at them. (Messaris, 1996, p.4)

Messaris does stress however that such visual imagery need not recreate real-world vision with complete accuracy, that even "cartoons, sketches, or black-and-white photographs" (1996, p.3) are capable of conveying the distinctive features required to elicit an emotional response.

This is the reason for the emphasis on visuals in both the industry and amongst the community as a whole; visual advertising is the most powerful and effective form of advertising. While the individual experience can vary when playing the game, the collective experience is much more consistent when faced with marketing. With this fact established, it can now be said that there is always justification for continued research into the visual aspect of computer games, and of the varying methods that can be used to produce aesthetically pleasing visuals for the sake of the collective experience.

3.2.0 – The evolution of style: How technology has had a direct influence on visual style.

Photorealism wasn't always the championed style however. In the past, the majority of computer games had a more stylised presentation. As the technology powering these games evolved, the attitudes towards computer game graphics shifted a fact that will be explored in more detail below.

3.2.1 – An introduction to the "box".

Figure 1: *Star Wars Rebel Assault* [Sega CD] (Star Wars, 2012)

When rendering technology was in its infancy, computer game visuals were simplistic and abstract. Gaming systems such as the Commodore 64 and the Odyssee were not capable of either storing or rendering a large amount of visual data (such as colour information or rasterised graphics), making it difficult for artists to produce detailed imagery. Jeremy Price, the CG Art Manager of Red Jade Studios, refers to these technical constraints as a metaphorical "box". Within his opinion piece written for Gamasutra, Price (2006) describes how artists used to work within this "box", "pushing their resourcefulness and their creativity to achieve the best results possible without breaching its borders."

During the early stages of computer game development, when this "box" was still small, the technology made it difficult for developers to create photorealistic games. Such early attempts resulted in games that were either aesthetically displeasing, or limited in interaction.

Star Wars Rebel Assault (Lucas Arts) for the Sega CD *[fig 1]* is an example of the former. Although the game uses digitized footage from the original movies, the footage has been compressed in order to fit within the console's specifications, leading to a noticeable drop in visual quality.

Figure 2: *Chrono Trigger* [SNES] (Chrono Trigger, n.d.)

Figure 3: Modern independent title VVVVVV [PC] (KnucklesSonic8 [pseud.], 2012)

Those that were limited in interaction include Full Motion Video (FMV) games such as *Mad Dog McCree* (American Laser Games) and *Night Trap* (Digital Pictures). Although these games looked no different to a typical TV at the time of their release (using live action footage of actors to tell the story), they didn't play like other games at the time, and instead were more like interactive movies. In the case of *Mad Dog McCree*, you had no control over where your character moved, nor could you choose which weapon to use. Instead, you watched a pre-recorded sequence of actors attacking you, pressing fire at the correct time in order to progress. Failing to press fire at the right time would stop the video, subsequently cutting to a new video of you being killed. As this was pre-recorded footage, enemies could only be shot in a specific order, as otherwise the game would require multiple pre-recorded sequences for every potential outcome. This leads to the game being more a question of remembering a pattern (like the electronic game *Simon*) as opposed to a question of skill. In short, FMV games played more like interactive movies than typical games due to their reliance on pre-recorded footage for photorealism.

As photorealism had proven itself to be both incredibly inefficient and difficult, developers instead were forced to find alternative ways to make their games aesthetically appealing. This resorted to many older games adopting a much more abstract, cartoon style *[fig 2]*. This style (colloquially referred to as "retro") is one that many designers revisit, being mimicked in contemporary titles such as *Super Meat Boy* (Team Meat), *Minecraft* (Mojang), and *VVVVVV* (Nicalis)*[fig 3]*. The "box" forced developers to come up with creative solutions to many of the design problems they encountered. One of the most famous of these solutions took place during the creation of arcade classic *Donkey Kong* (Nintendo). According to an article written for the Official Nintendo Magazine (East, 2010), the now iconic design of Mario was a solution to a series of limitations imposed by the technology they were using. Shigeru Miyamoto, the creator of Mario, had only a 16x16 pixel grid for which to design and animate him, as well as a limited colour palette. As it was impossible for Miyamoto to animate Mario's hair in a convincing way, he decided to give Mario a hat so that he wouldn't have to. As for the dungarees, Miyamoto wanted players to be able to clearly view Mario's running animation, another thing that was difficult to accomplish within the design restraints. Miyamoto needed a design that would allow him to use colour to clearly distinguish Mario's arms from his chest, and for this he decided to use dungarees. This design is in continued use today, with modern Mario titles utilising this same iconic design despite the aforementioned limitations no longer being an issue. *[fig 4, right]*

In the interests of transparency, it should be noted that the Official Nintendo Magazine (East, 2010) incorrectly attributed this information to *Super Mario Bros* for the NES (Nintendo Entertainment System), produced four years after Mario's first appearance as 'Jumpman' in *Donkey Kong*.

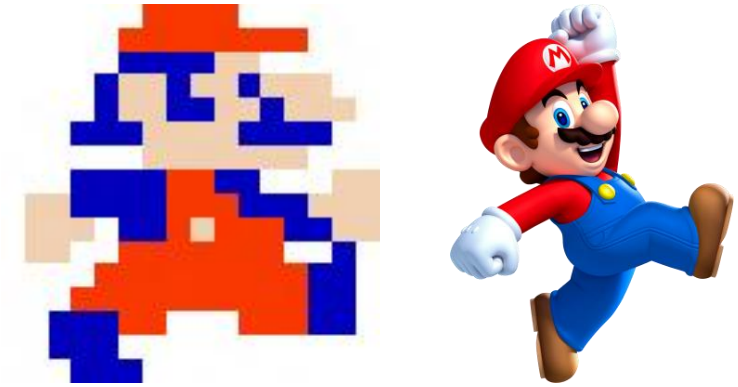


Figure 4: (Left) Mario as he appears in *Donkey Kong*. (Locke_GB7 [pseud.], n.d.)
(Right) Mario as he appears now. (Prince Ludwig [pseud.], 2012)

There have been various other cases where the need for creative solutions to issues imposed by the "box" resulted in games with unique visual styles and effects. Such examples include *Soul Blade* (Namco), which relied heavily on colour usage and dynamic lighting to set the atmosphere of each level; *Fear Effect* (Eidos), with its comic book aesthetic; *Resident Evil* (Capcom), which made up for its static, pre-rendered backgrounds with the use of atmospheric camera angles, a technique which became a staple of the franchise; *Tomba!* (Whoopee Camp), with its vibrant cartoon characters and eccentric worlds; and *Silent Hill*, who's iconic thick fog was due to the engine's short drawing distance.

3.2.2 – The "box" grows.

Throughout the years the "box" has grown, with there being a great deal of advancement in the technology used to render computer games, including:

- The adoption and subsequent improvement of colour graphics. Originally games were rendered in monochrome, with some arcade machines using sheets of coloured plastic to fake the impression of colour graphics. This changed in 1974 however with the release of Namco's *Galaxian*, the first full-colour video game. (Donovan, 2010, p.85) At first the available colour palette was limited, but this improved with technological evolution.

- The transition from 2D sprite art to 3D polygonal rendering, a process which in itself required various smaller steps to achieve. The earliest examples of 3D games used wireframe visuals to give the impression of 3D graphics within a 2D space. This method was popularised by games such as *Battlezone* (Atari). Atari then introduced the use of coloured polygons to video games with the release of *I, Robot* in 1983. This was then subsequently improved upon by id Software with *Catacomb 3-D*, the first 3D game to feature texture-mapped polygons, adding additional detail to the previously flat coloured polygons. This utilised a method known as raycasting, which was subsequently used in other idSoftware games, most notably *Wolfenstein 3D* and *Doom*. (Donovan, 2010, pp.250-263) Further improvements to 3D rendering included the introduction of OpenGL (Open Graphics Library) and Microsoft's DirectX libraries.

- The improvements made to the aforementioned graphics libraries opened up a collection of new features that allowed designers to further improve the appearance of 3D scenes, including the use of shaders. Shaders allow the artist to affect how a particular surface looks

by changing the way it's shaded. For example, shaders could now be used to make some surfaces look like rippling water, plastic, skin, hair, glass, or a painting, by utilising methods like subsurface scattering, anisotropic specular, and cubemaps.

- The increase in available screen resolutions (i.e the size of the image shown). The NES had a rendering output of 256x240 pixels. The Playstation supported resolutions up to 640x480. In the present day, it is expected for games to support at least a resolution of 1280x720 (720p) if not 1920x1080 (1080p). Higher resolutions can also be achieved on a PC, as well as on other particular devices (such as the third iPad). Not only does the increased resolution mean that more detail can be viewed on the screen, it also provides numerous other possibilities for an immersive experience. For example, it is possible to play a racing game with the display stretched across multiple screens, allowing for both the front and sides of the car interior to be viewed at once. With these screens positioned around the viewer, it can provide them with an immersive experience.

- The introduction and the further improvement in physics simulations. Although physics has become a large part of modern day gameplay (with games like *Peggle*, *Garry's Mod*, *Limbo*, and *Half Life 2* featuring physics as major gameplay elements), the evolution of physics technology has also had a significant visual impact. Modern day physics engines allow developers to create characters with hair and clothing that trails realistically with their movements, helping to provide visual feedback on elements such as how fast the character is moving, or how strong the wind in a particular area is. Even primitive physics engines such as the ones used in *Tomb Raider 2* (Core Design) or *Virtua Fighter* (Sega) help to make characters appear more dynamic. The same can be applied to locations too, with physics being used for falling debris, splashes of water, spurts of blood, and smoke, amongst other things.

- The improvements in hardware, especially space and memory. Without the space to store all of the data required, or the memory needed for the final rendering, all of the previous advancements would be a moot point. *Crysis* (EA) in particular requires a large amount of memory in order to run with the highest graphical settings without crashing, more memory than what is available on any 32-bit system (hence the inclusion of 64-bit binaries).

As the "box" has grown, artists have found themselves capable of producing more visually detailed work. It has however now come to the point where artists are no longer working within constraints, but are now actively trying to push constraints. As noted before in section 3.1.0, this process can be observed through the many tech demos that have been produced, marking each step in the evolution of technology. This phenomenon has been witnessed and noted by Price (2006) in the aforementioned opinion piece.

> With the latest generation of hardware, the box is now bigger than it has ever been before. Yet it is at this point in history that a fundamental change has taken place. For the first time ever, 'technical feasibility' is no longer the major limiting factor in what can be achieved visually. (Price, 2006)

3.3.0 – The photorealism bias.

By observing the tech demos that have been released within the last decade, the ultimate goal behind this evolution of technology is evident; photorealism.

Dawn (NVidia), *Adrianne* (NVidia), *Nurien* (Nurien Software), *The Final Fantasy VII PS3 Tech Demo* (Square Enix), *The Casting* (Quantic Dream), *Medusa* (NVidia), *Kara* (Quantic Dream), *Unlimited Detail* (Euclideon), *Agni's Philosophy* (Square Enix), *Samaritan* (Epic), *Elemental* (Epic), and *New Dawn* (NVidia) are just a few examples of well known tech demos produced by both game companies and hardware developers, all of which focus on the ability to render photorealism.

Reading through the official article written for the *New Dawn* demo by NVidia (Wang, 2012) provides a much closer insight into the effort and focus put into the rendering of photorealistic graphics. It describes in great detail how technology like Tesselation allowed them to give Dawn realistic strand-based hair that would flow in the wind, rendered to look "soft and silky, as if she just jumped out of the shower after an extensive conditioner routine". There is also a particularly large amount of focus put into describing how skin in the real world works in accordance to lighting, and how subsurface scattering was used to render skin that wasn't "too glossy" or "too matte", but "actually looks right". Finally, the article describes how post-process shading was used to apply additional effects onto the screen. These effects were not used to simulate how such a scene would appear to human eyes, but instead through a real world camera through the use of bokeh depth of field (bokeh being a form of photography originating from Japan). Such methods are not unique, as many other recent demos showcase the same technology and methods.

One demo however that stands out in particular is the *Kara* tech demo by Quantic Dream, an emotionally driven piece about a female cyborg who, unlike other models in her line, has developed human emotions. Unlike most other tech demos, *Kara* has a fleshed out storyline, likening itself to a short movie as opposed to an elaborate display. Also unlike the aforementioned *New Dawn*, the official writeup for this demo concentrated much more on the character's behaviour, with emphasis on high quality motion capture to "improve the quality of the acting" (Shuman, 2012). Even so, it is clear that their goal is still the same; to provide the most photorealistic experience (in both visuals and acting).

By observing these tech demos, it can be ascertained there is an emphasis within the mainstream computer games industry to strive towards the goal of photo-realism. As the "box" continues to expand, artists within the industry have begun to strive for this goal of photorealism more than ever before, putting in more emphasis into research that makes this goal possible.

Now, while it can be argued that at this point in time there appears to be just as many non-photorealistic games as there are photorealistic games, a clear distinction needs to be made between the mainstream and independent industries. The majority of contemporary stylised games, including titles such as *Minecraft* (Mojang), *McPixel* (SoS), *Home* (Benjamin Rivers Inc.), *Deponia* (Daedalic Entertainment), *Braid* (Number None, Inc.), *Limbo* (Playdead), *Hotline Miami* (Dennaton Games), *Bastion* (Supergiant Games), *Giana Sisters* (Black Forest Games), and *Sword & Sworcery* (Superbrothers), are produced by independent game development groups, many of which attempt to emulate the 'retro' style as noted before in section 3.2.1. Very few independent games (henceforth referred to as indie games) are rendered in a photorealistic style. Instead, most of the

photorealistic games published at the writing of this thesis are produced by the mainstream industry, the same industry that produces these tech demos. Steph Greenberg (1999) has noticed the same strive for photorealism in the CGI (computer generated imagery) industry, noting in her submission to Siggraph:

> Since computer graphics' inception, programmers and artisans in computer animation have sought increasing realism in rendering, surface representation and movement. The "holy grail" of computer animation would appear, to an outside observer, to be an image indistinguishable from the reality we all experience. (Greenberg, 1999)

3.4.0 – Should photorealism be the ultimate goal?

3.4.1 – Photorealism's place in computer games design.

This emphasis on achieving photorealism will most likely be detrimental to the future of computer games design. As more companies choose to follow this common goal and utilise photorealism in the majority of their products, this will lead to games becoming visually unremarkable. In its purest form, there is only one style possible with photorealism, and that is photorealism itself. There is only so much artistic license that can be taken with the real world before it ceases to look like the real world, thereby nullifying the point of photorealism. Even a photograph can cease to look like a believable depiction of the real world through the use of photo editing. In summary, all games that achieve the goal of true photorealism will all look alike, bearing no unique visual elements to set themselves apart.

This is an opinion shared by Sugano Yoshinori (1999) who, on the subject of photorealistic computer graphics (CG) imagery used in television shows, stated that such CG "tends to look monolithic and excessively clean-cut". He continues to observe that "the pervading opinion of late seems to be that CG should be used such that an audience can't detect its presence – a mundane goal in my opinion."

This isn't to say that photorealism doesn't have the potential to be visually appealing, or that there isn't a place for photorealism within video games. Games such as *Heavy Rain* by Quantic Dream benefit from the use photorealistic graphics. *Heavy Rain* manages to create the experience of a truly interactive movie by successfully digitizing not just the physical appearance of, but also the performances of real actors, providing a familiar and empathetic experience. With these techniques, *Heavy Rain* manages to emotionally engage the player, forming an empathetic connection between them and the characters. As shown by the *Kara* tech demo discussed back in Section 3.3.0, this is clearly a philosophy that Quantic Dream truly believes in, wanting to improve their methods in order to bring a much more photorealistic and, consequently, "real" experience to the player.

Photorealism, however, shouldn't be considered the "holy grail" (Greenberg, 1999) of games design. It should instead be treat only as a style; one of many styles. Just like how *Heavy Rain* was better suited to the photorealistic style, there are many games that are better suited to other styles, and as such all of these potential styles should be treated with the same drive and enthusiasm that photorealism currently receives. At the time of writing however, this isn't the case.

3.4.2 – The lack of variety.

Non-photorealistic rendering is capable of producing a wide range of interesting imagery, a fact that has been proven with the continued academic research by those participating in the International

Symposium on Non-Photorealistic Animation and Rendering (NPAR), an annual symposium dedicated to the research of non-photorealism. The many academic papers produced by the symposium include works such as the paper on “Real-Time Watercolor Illustrations of Plants Using a Blurred Depth Test” by Thomas Luft and Oliver Deussen of the University of Konstanz, which shows how their methods can be used to create convincing watercolour renders of fauna in real-time. Such methods are rarely utilised in a gaming environment, however, in favour of much more simplistic and common methods. This is a phenomenon also observed by Doug Cooper (1999), noting that most of what he has observed in terms of non-photorealistic rendering centres around “toon-shading”, potentially giving the impression that “this is NPR’s only current application area.” (Cooper, 1999) He goes on to note that non-photorealistic rendering appears to be used less for its potential to look good, but simply as an easier and faster way to produce art. “We use these tools because they produce a look that is ‘good enough’”. (Cooper, 1999)

4.0.0 – Deciphering the reasons for NPR's absence.

4.1.0 – PR versus NPR: Hypothesis

While the symposium mentioned in section 3.4.2 has proven many times that a variety of interesting visuals are possible with non-photorealist rendering methods, this seems to have had no effect on the prominence of photorealism. As such, a lack of knowledge or research into this department cannot be the blame for NPR's absence, and the issue must lie elsewhere. As technology has improved the photorealism style has become more and more prominent (a fact detailed in section 3.2.0). This isn't the only change that has taken place however; games in general have also become more popular, gaining a much wider market. For this reason, it is possible that this is an issue of trend; photorealism is utilised more as it is the more popular style amongst consumers.

4.2.0 – Existing Theories

There are some pre-existing theories that, upon first glance, support this hypothesis.

4.2.1 – Immersion.

One of the existing theories for this prominence of photorealism is due to its contribution to immersion. Immersion can be described in one of three ways; flow, cognitive absorption, and presence. (Jennett, 2008)

Out of these three forms of immersion, the third (presence) is the most important. Slater, Usoh & Steed (1994, as cited by Jennett, 2008) define presence as "a psychological sense of being in a virtual environment". This is when the individual feels as though the fictional world they are experiencing is just as living and breathing as the real world. This is the form of immersion most commonly referred to in the context of open environment games, such as role playing games or first person shooters, with developers wishing players to be able to feel as though the worlds they are exploring are real.

A stronger sense of presence is achieved when the virtual environments and the interactions within "mimic real-world experiences" (Jennett, 2008). This sense of realism can be split into two categories; social realism, and perceptual realism. Social realism describes how closely the interactions/behaviour in the virtual world matches the real world. Perceptual realism describes the closeness in physical appearance. (McMahan, 2003, p.75) The combination of the two of these can help to dramatically increase the feeling of presence.

Taking this into consideration, it could be understood why a company wishing to make a truly 'immersive' game may choose to use a photorealistic style, taking advantage of the heightened perceptual realism to increase the feeling of presence. This line of thought has been utilised in many of the more recent open world games that rely on presence, such as *Skyrim* (Bethesda), *Grand Theft Auto 4* (R*), *Sleeping Dogs* (Square Enix), and *Hitman: Absolution* (Square Enix).

While this might be a sound assumption, it is a disproven one. According to McMahan (2003, p.68), most scholars and scientists agree that photorealism isn't necessary for the player to feel immersed. Although the combination of both social and perceptual realism can be used to heighten realism, both are not necessary; as noted by Clive Fencott (1999, as cited by McMahan, 2003), a cartoon styled world may have a low degree of perceptual realism, but could potentially have a high degree of social realism.

Furthermore, it should be noted that photorealistic games have a much higher chance of evoking the uncanny valley. The uncanny valley is described by Mori (1970) as being the negative reaction to something that appears real but lacks familiarity. Within the context of robotics and prosthetics, this can be compared to a prosthetic hand, one that looks real to the point where there is equal expectation for the prosthetic to share the same traits as a real hand (such as temperature, or the feeling of soft tissue). When these expectations are broken, the subject may experience a feeling of unease; "[…]this kind of prosthetic hand is too real and when we notice it is prosthetic, we have a sense of strangeness." This negative feeling can be enhanced with the introduction of motion. (Mori, 1970)

In the context of a video game, the uncanny valley may refer to a character that looks realistic but does not move or behave realistically, either due to the limitations of the engine or the limitations of the artist. A realistic looking character sets the expectation that the character should also move and behave just as realistically. Should this expectation be broken by common issues such as inadequate facial animation, this may produce the negative experience of the uncanny valley. Such cases may have the effect of either unnerving or distracting players from being able to suspend their disbelief, thus breaking the feeling of immersion. Stylised games such as *Elite* (Acornsoft) or *Limbo* (Playdead) do not suffer from this, as their appearance does not set up any expectation for the characters to move or behave in accordance with real-world expectations.

4.2.2 – A cartoon stereotype.

Another possible reason for the unpopularity of non-photorealism is due to its resemblance to children's media, specifically cartoons. As noted by Mitchell (1995), cartoon media has traditionally been regarded as "kids' stuff" that is unworthy of attention among adults.

> Once, animation held a strong position in our society, as people of all ages and walks of life enjoyed cartoons in the movie theatre and on their television sets. The humor of the early cartoons was aimed at audiences of all ages and sophistication levels. In recent decades, however, cartoons lost their cross-generational appeal. (Mitchell, 1995)

This is similar to what has happened with video games, as discussed in 3.2.0. Although cartoon games were originally aimed at a wide audience, over time these games have been shunned in favour of more realistic ones. Furthermore, there has been a long standing assumption by the mainstream public that videogames were predominantly for children, a belief that powered many a public outrage over videogame violence. For example, when the videogames *Mortal Kombat* (Midway) and *Night Trap* (Digital Pictures) were first released, their violent content shocked mainstream audiences who believed that such games were "corrupting the nation's children". (Donovan, 2010, p.225)

With this in mind, it can be understood that both gamers and developers alike may wish to break away from this image. One of the ways to do this would be by distancing themselves from an art style that makes games look like "kids' stuff", and focusing more on the grittier and more mature style of photorealism.

In spite of this however, there exist games such as *Borderlands* (Gearbox), *Zone of the Enders* (Konami), and *Walking Dead* (TellTale), all of which are stylised games aimed at older audiences. Not only do such games exist, however; such games are also successful. Based on each game's Metacritic

scores at the time of writing this, all of them proved to be just as critically successful as their closest photorealistic counterparts at the time. For example, *Borderlands* received a Metacritic score of 81, while *Call of Duty: Black Ops II* (Activision) received a score of 83; *Zone of the Enders* received a score of 78, while *Steel Battalion* (Capcom) received a score of 83; finally, *Walking Dead* received a score of 89, while the similarly themed *Resident Evil 6* received a score of 74. As such, the belief that NPR cannot be used in games intended for older audiences seems flawed.

4.2.3 – The apparent anime stigma.

While both of these theories are flawed, the latter theory (section 4.2.2) can actually be observed within the Japanese game industry, specifically in the difficulty many stylised Japanese games endure when being released in western territories. It is typical of a game to have changes made to it as it is released in a different territory. These changes are usually centred on the language barriers of each territory, and may include the translation of any spoken and/or written dialogue in the game, and the removal or editing of puns, jokes, and cultural references that wouldn't make sense in other languages or cultures (for example, the numerous puns in *Phoenix Wright: Ace Attorney* by Capcom).

There have been some cases however, where not only has the language been changed to suit a different territory, but parts of the visual style too. Such cases include *Agarest Senki*, *Star Ocean: The Last Hope*, and *Megaman*.

Figure 5: (Left) The final box art for *Agarest Senki* utilising the original artwork. (Agarest, n.d.) (Right) The proposed European box art. (Artefact [pseud.], 2009a)

Agarest Senki, developed by Idea Factory, is a Japanese Role Playing Game (JRPG) for the Playstation 3. It is known as *Agarest: Generations of War* in Europe, and *Record of Agarest War* in America. The game's visual style is incredibly stylised, featuring flamboyant character designs and bright colours, all presented with a typical anime aesthetic. This style is continuous throughout the packaging, promotional material, and in-game graphics. *[fig 5, left]* During the localisation process however, publisher Ghostlight revealed plans to redesign the European packaging of the game with a much more photorealistic style, featuring a much older and more aggressive version of the protagonist. *[fig 5, right]* The new design was met with derision by the fans, which led them to use the original artwork instead. (Tolentino, 2009)

Figure 6: (Top) The Japanese version of *Star Ocean: The Last Hope*. (Bottom) The English version of *Star Ocean: The Last Hope*. (Artefact [pseud.], 2009b)

Star Ocean: The Last Hope, (hereafter referred to as *SO:TLH*) the fourth in the series, is a JRPG available on both the Playstation 3 and the Xbox 360. *SO:TLH* went through a very similar process to *Agarest Senki*, except with the in-game artwork instead of the box art. In the game, profile images of the characters in your party are used in both the Battle and Menu interfaces, allowing the player to easily view at a glance which character's information they are looking at. In the Japanese version of *SO:TLH*, these profile images are drawn in a 2D anime style. *[fig 6, top]* In both the American and European versions however, this artwork has been replaced with realistic CGI renders of each character. *[fig 6, bottom]* (Spencer [pseud.], 2009)

Figure 7: The Megaman box artwork for the; (Left) American (Mega Man Box US, 2010), (Middle) Japanese (Rockman Box JP, 2010), (Right) and European versions. (Mega Man Box EU, 2010)

This is not a new trend, however. Anime themed artwork has been replaced in western releases of games since the late 80s, as can be seen with the *Mega Man* franchise. Named *Rockman* in Japan, *Mega Man* is a game produced by Capcom that features the game's namesake (an android) making his way past various enemies and obstacles in order to defeat the evil Dr.Wily, all presented as a 2D platformer. The original Japanese box-art *[fig 7, middle]* shows the main characters of the game drawn in a typical anime style with cartoon proportions, giving them a cute appearance. The American box-art however is completely different *[fig 7, left]*, portraying the character of Mega Man as a middle-aged human with a handgun (as opposed to an arm that changes into a gun, as in the game). The cover itself is poorly produced and very inaccurate to the game, and has since become notorious amongst *Mega Man* fans (to the point that a character based off this design was included in *Marvel Vs Capcom 3*). The European version of the box-art *[fig 7, right]*, while much more accurate than the American box-art when portraying Mega Man's design, still takes a very different approach to portraying the game as a whole. While the original Japanese box-art portrays the game as something flamboyant and simplistic, the European box-art attempts to portray the game as a much darker and more serious game, and is again drawn in a much more realistic style.

It is possible that such changes were made due to the belief that the American audience prefers darker and more realistic games over more vibrant stylised games, potentially to the point where such stylised games are ostracised by the industry. On their official Facebook page, Xseed (a company that specialises in localising Japanese games such as *Zwei*) responded to a fan's question

about their work with “retailers aren’t too crazy about the cutesy graphics, which makes our job a little more difficult.” (XSEED Games [pseud.], 2010)

A similar line of thought is shared by Square Enix, producers of the renowned *Final Fantasy* series and the aforementioned *Star Ocean* series. This can be observed in some of the design decisions made during the creation of *Nier*, an action game released on both the Playstation 3 (PS3) and the Xbox 360. While they didn’t change the art style, which is realistic by default, they did change the design of the main character between the two versions; *Nier Replicant* and *Nier Gestalt*. The original *Replicant* version, exclusive to Japan (PS3), features a younger and more effeminate male lead. *[fig 8, right]* The *Gestalt* version on the other hand, released in Japan (Xbox 360), America, and Europe (PS3 and Xbox 360 as *Nier*), features a much older and more muscular male lead. *[fig 8, left]* In an interview with Inside-Games (translated into English by Artefact [pseud.], 2010b), director Taro Yoko explained that the change was due to the concern of international markets. They felt that a version of the game featuring a youthful protagonist would not do well in the international market. “For the North American consumers, it was decided to provide a macho main”. (Artefact [pseud.], 2012) This is very similar to the aforementioned redesigns of *Mega Man*; not only was the art style of *Mega Man*’s box art changed, but the design of Mega Man himself was altered to appear older and manlier.

Figure 8: The protagonist as he appears in (Left) Nier Gestalt (Heath [pseud.], 2010a), and (Right) Nier Replicant (Heath [pseud.], 2010b).

4.3.0 – Experiment #1: Consumer Survey.

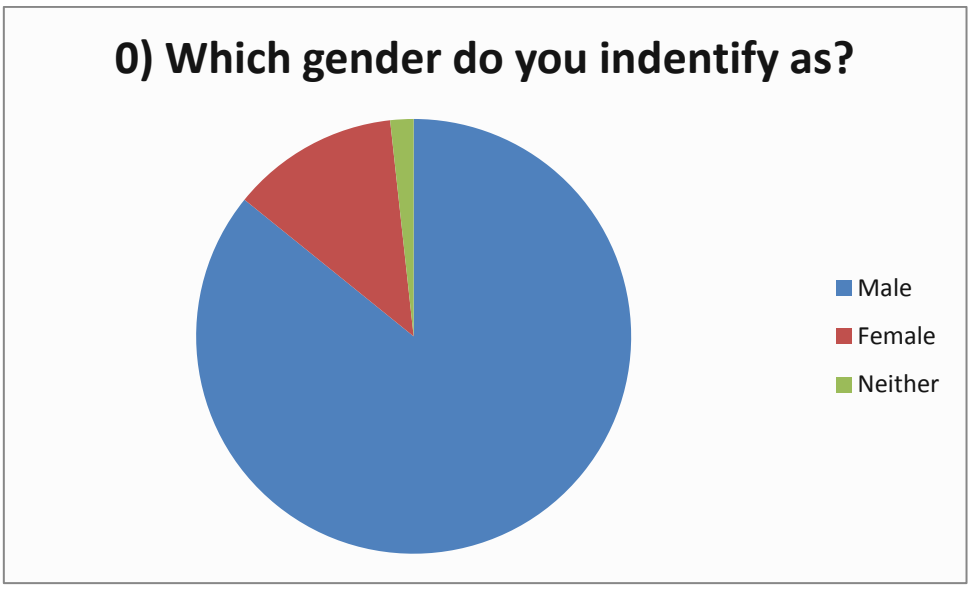
As can be observed from the prior examples, there is an inherent belief within these companies that any game straying too far from a grittier and more realistic presentation will be less financially successful in particular territories. This can be considered evidence proving the theory explained in section 4.2.2, which in turn can be considered evidence supporting the hypothesis in section 4.1.0.

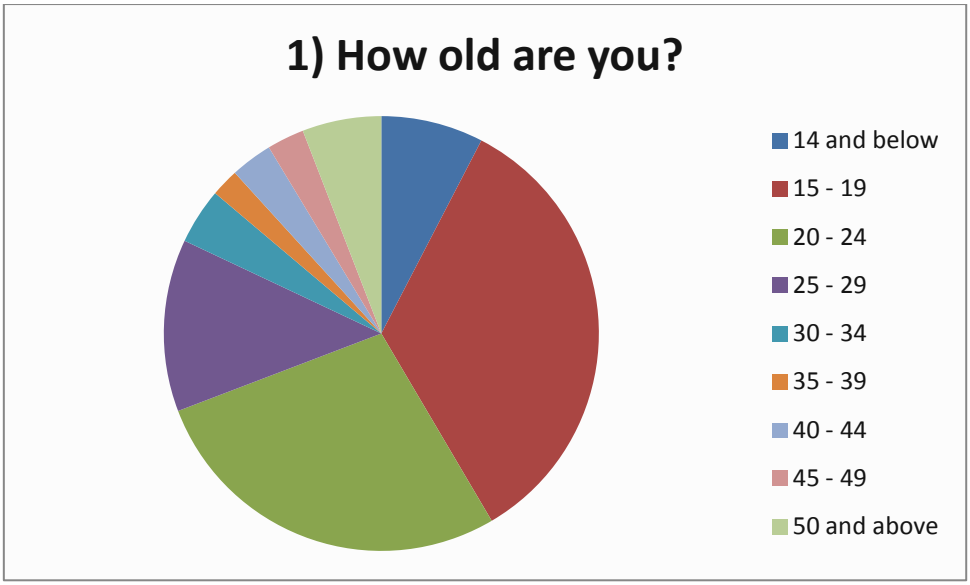
This is still inconclusive, however. At this point, all that has been proven is the fact that the companies themselves believe in the cartoon stigma, not that it works in practise. The only way to certify whether either of the existing theories in section 4.2.0 apply to the consumers is to ask the consumers themselves. As such, a survey of over 300 people across a multitude of gaming communities has been conducted asking them about their feelings towards photorealism and non-photorealism, the results of which can be seen below. This survey has been conducted over the internet to ensure a wide variety of answers from all over the world.

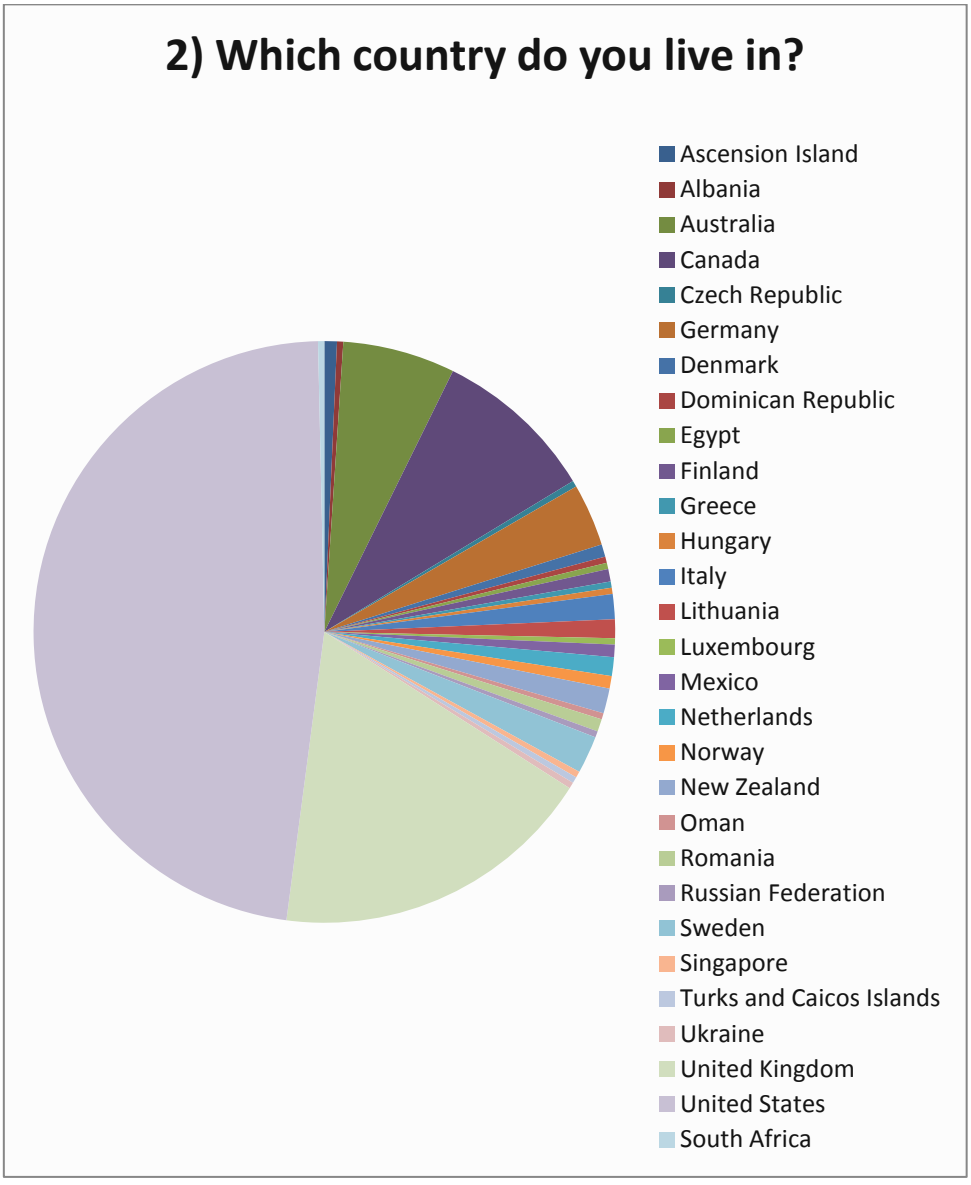
To reiterate, the hypothesis (as laid out in section 4.1.0) is that photorealism will prove to be the more popular style amongst consumers, therefore justifying the gaming industry's focus towards photorealism.

4.3.1 – Survey Results.

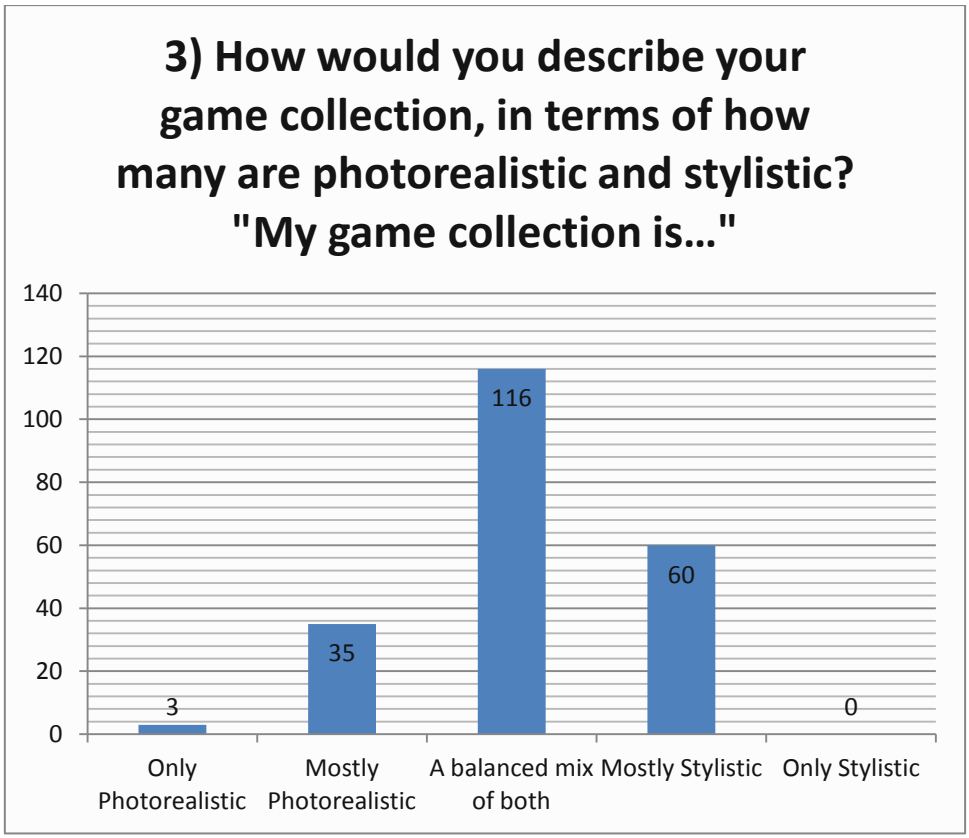
Questions 0 – 2 are typical demographic questions, included to both prove the variety of participants and to see if there are any patterns in responses depending on demographic.

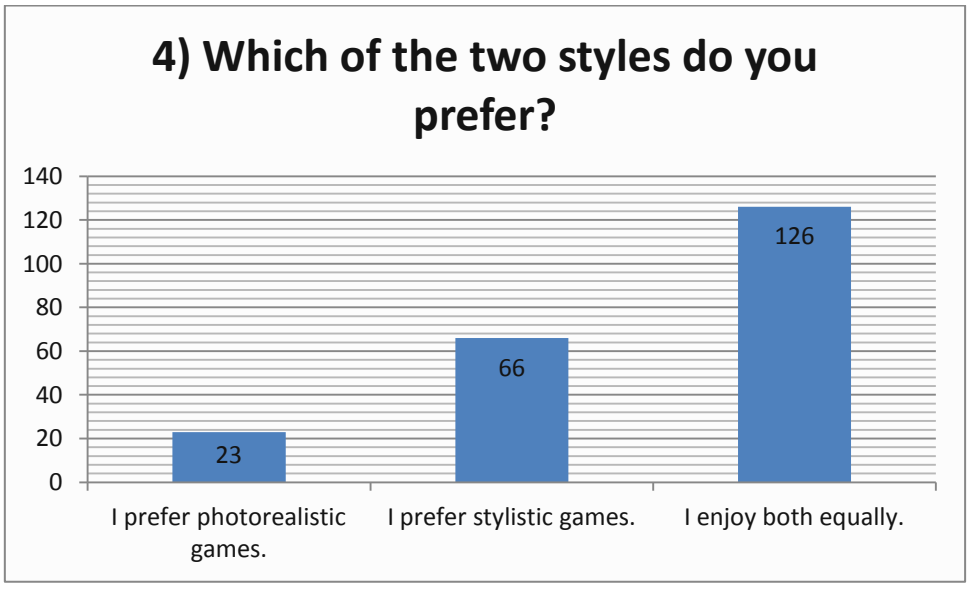



1) How old are you?
14 and below
15 - 19
20 - 24
25 - 29
30 - 34
35 - 39
40 - 44
45 - 49
50 and above


2) Which country do you live in?
Ascension Island
Albania
Australia
Canada
Czech Republic
Germany
Denmark
Dominican Republic
Egypt
Finland
Greece
Hungary
Italy
Lithuania
Luxembourg
Mexico
Netherlands
Norway
New Zealand
Oman
Romania
Russian Federation
Sweden
Singapore
Turks and Caicos Islands
Ukraine
United Kingdom
United States
South Africa

Questions 3 – 5 are to gauge the popularity of photorealism and non-photorealism amongst the participants.



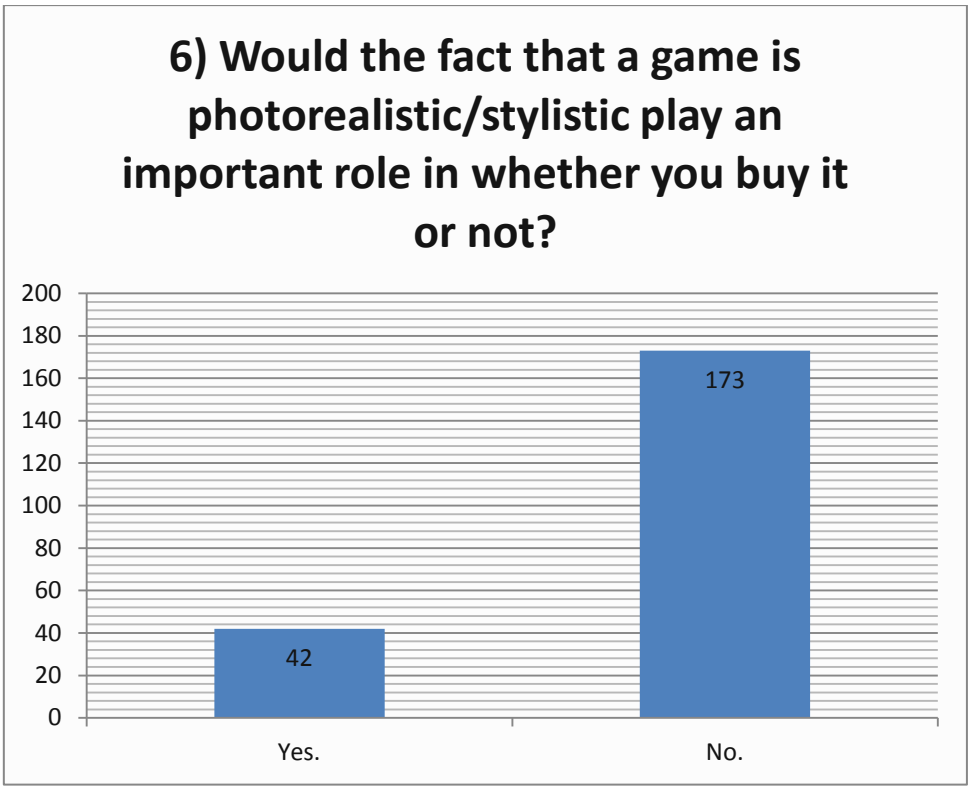


5) Why?

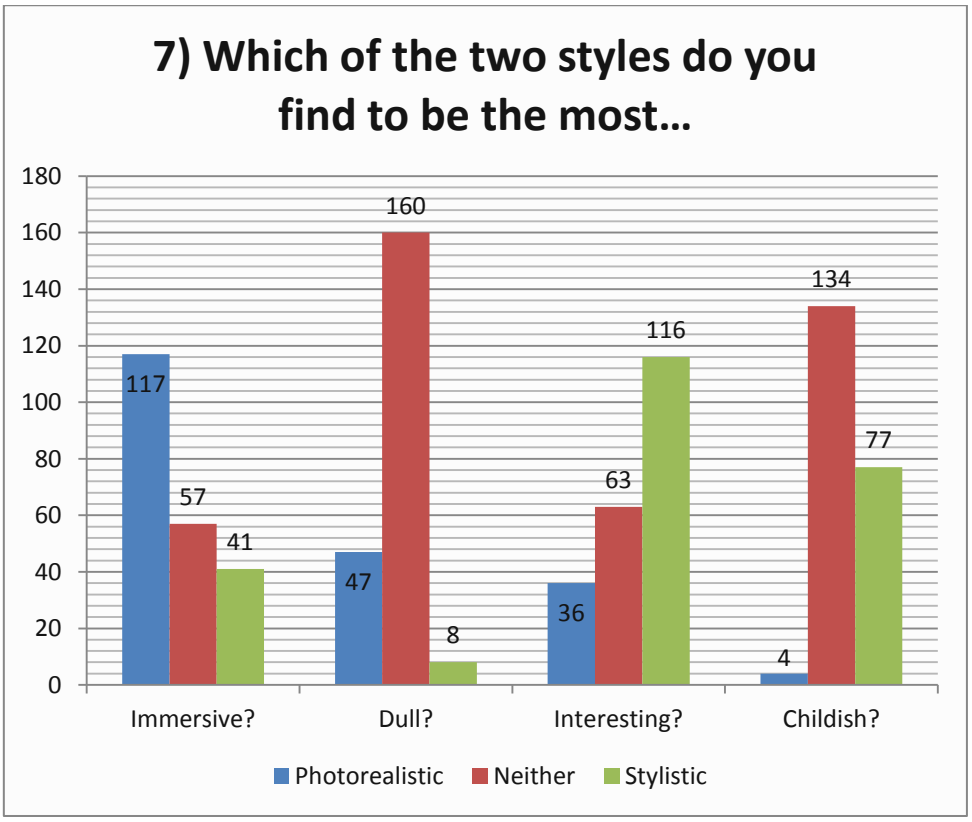
(You can be as simple or as thorough as you please.)

For this question (5), participants were required to write their own answer.

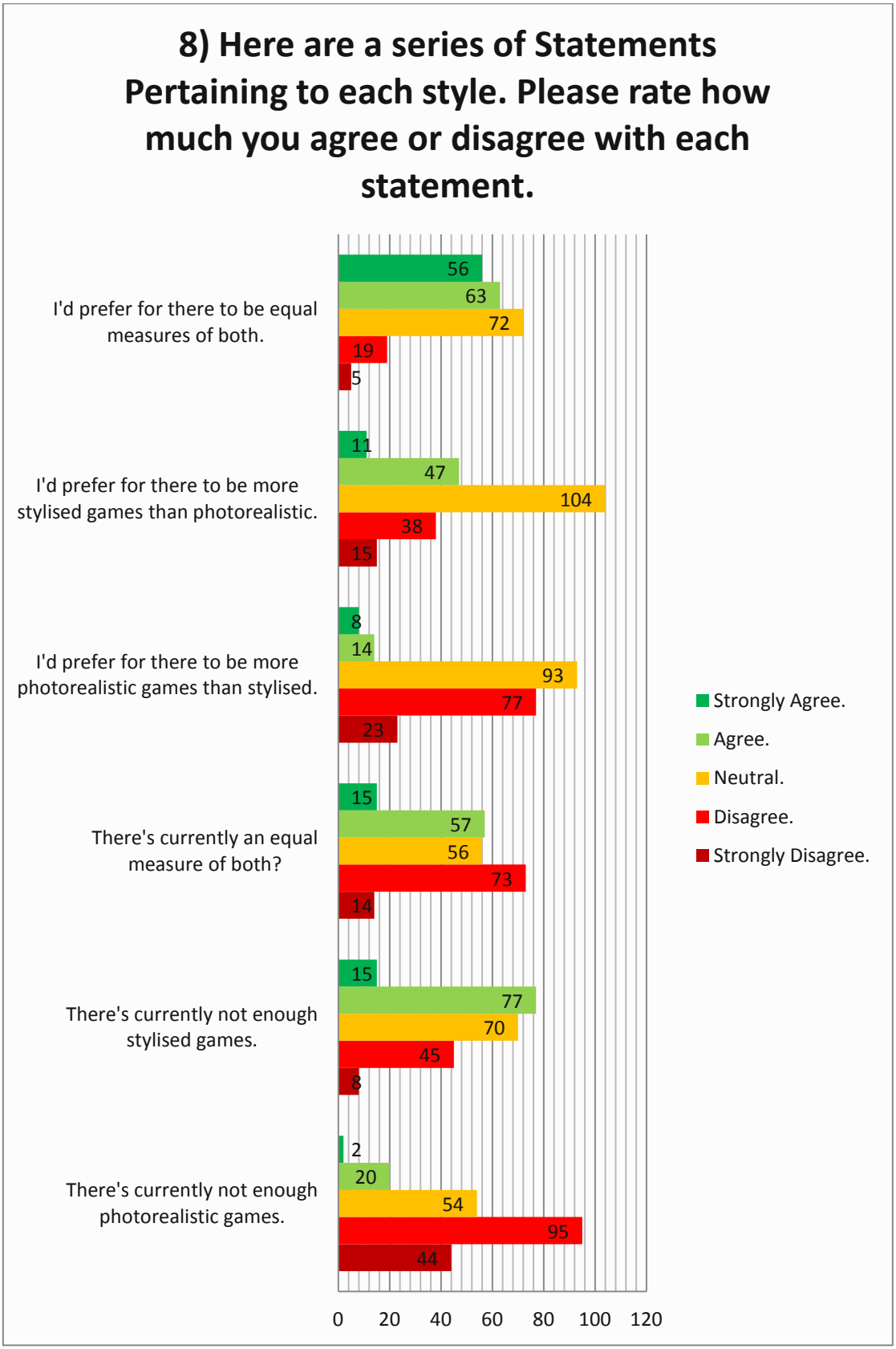
The point of question 6 is to help gauge whether the cartoon stereotype (section 4.2.2) or the anime stigma (section 4.2.3) are genuine concerns.



Question 7 is a simple word association question based of the prior theories in sections 4.2.1 and 4.2.2.



This question (8) is to help gauge the consumer's opinions on the current games available in a way that is visually simple. At a glance, it is easy to tell which responses are strongly positive, negative, or apathetic.

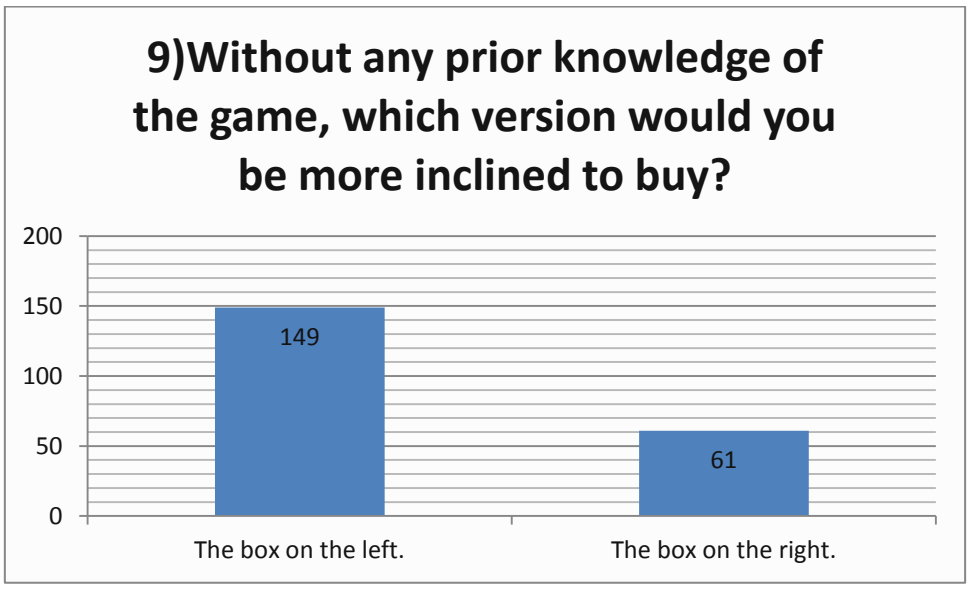


Questions 9 – 11 were directly based on one of the examples of the anima stigma in section 4.2.3. The participants were provided with the following image and explanation.

[Figure 5, Section 4.2.3]: Left (Agarest, n.d.) Right (Artefact [pseud.], 2009a)

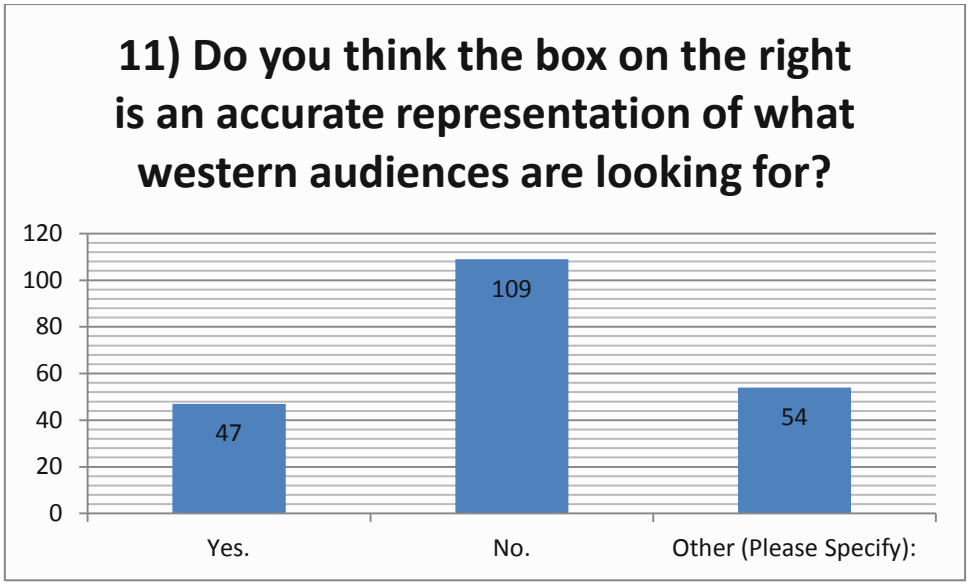
"The above image shows two box covers for the Japanese Playstation 3 game Agarest: Generations of War.

On the left is the final design for the American release, utilising the original Japanese artwork. On the right was a proposed design for the American release, created with the intention of appealing to an American audience. The design features an older and more masculine version of the main character (seen on the bottom of the final cover), rendered in a realistic CGI style."



10) Why?
(Be as simple or as thorough as you want.)

For this question (10), participants were required to write their own answer.



Participants answering "other" to question 11 were required to write their own answer.

4.3.2 – Photorealism vs. non-photorealism: The observations.

Numerous useful observations can be extrapolated from this data.

- Based on questions **three** and **four**, the majority of participants do not have a particular preference over photorealistic or non-photorealistic rendering in computer games, and tend to like both in equal measure. This is further bolstered by question **six**, which shows the majority of participants do not feel the art style of the game to be a major factor in whether they buy it or not. This goes against the previous assumption that the art style of the game will affect the likelihood of it being successful (section 4.2.3). This alone already disproves the hypothesis in section 4.1.0; this cannot be a trend issue, as not only do the majority of consumers like both styles in equal measure (they do not feel as strongly about photorealism as some companies might assume), but the art style isn't even a notable deciding factor in which games they purchase.

- Although the art style of a game will have little effect on the games potential sales, these styles are interpreted in different ways, as shown by the text responses given to question **five**. Recurring themes amongst these answers include the opinion that photorealistic games are better at providing an immersive atmosphere (a point explained before in section 4.2.1). Conversely, the recurring themes among responses in favour of stylised games include words such as "unique" and "distinctive".

> photorealistic for games such as mgs and batman create the atmosphere and tension as best compared to movies and television. stylistic games such as Prince of Persia 2008 can create beautiful unreal worlds that cannot be recreated in real life therefore provide an insight into the surreal and impossible beauty. something that can only be provided by cartoons or dreams. [sic] (*Anonymous*, personal communication, 2012)

These two positive opinions of each style are further bolstered by question **seven** (the word association question), where once again photorealistic games are considered the most immersive, while stylised games are considered the most interesting.

- Question **seven** also disproves the cartoon stereotype theory explained in section 4.2.2, but only to an extent. While the majority of participants disagreed when faced with the suggestion that non-photorealistic games were childish, there is still a sizable amount of participants who agreed with this word association. While such an opinion is in the minority, it does at the least justify the feelings expressed by some companies (such as Xseed in section 4.2.3) about the misinterpretation of their games based on the art style.

- Two things can be determined from question **eight**. The first is that the majority of participants agree to some extent that there is a noticeable gap in the market for stylised games, disagreeing with the notion that there is currently an equal balance of both (and strongly disagreeing with the notion that photorealistic games are in the minority). The majority of the participants would also prefer a balanced mix of both photorealistic and non-photorealistic games. This is further evidence of the hypothesis in section 4.1.0 being wrong; photorealism's popularity is potentially exaggerated.

The final questions pertaining to the *Agarest* artwork also proved insightful. Most of the applicants who had chosen the original artwork over the new artwork hadn't done so due to a particular preference to the anime style, but had instead done so for other reasons. One of the reasons provided was the quality of the style. Some of the responses given stated that the photorealistic cover was poorly executed, with particular emphasis on the poor design of the character and the lacklustre composition. The other given reason pertained more to its inaccuracy in representing the game.

> Because the one on the right could be classed as false advertising, and the front cover should show you what you get in game, instead of a more western version just for the sake of appealing to western audiences more. [sic] (*Anonymous*, personal communication, 2012)

> The guy on the right looks like a cosplayer. His hairstyle does not suit his facial structure. His facial features also do not reflect the typical structure of a realistic looking heroic character. The more realistic you make the character, the harder it is to keep up the suspension of disbelief that such a person would wear such a ridiculous look and still maintain any form of seriousness. [sic] (*Anonymous*, personal communication, 2012)

Those who chose the original artwork assumed that, as it was the original artwork, it was an accurate representation of the in-game art style. Some took the initiative to confirm this by briefly researching the game and looking for screenshots. These participants took issue with the fact that the photorealistic cover was an inaccurate representation of how the game itself actually looked, with some going as far as to call it a form of false advertising. This further emphasises the previous point of consumers being less concerned about the art style of the game, and more about the experience of the game.

Finally, when asked if the photorealistic artwork was an accurate representation of what the majority of western consumers want, the majority believed this to be false. It should be noted

however that a large amount of participants used the "Other" category as a way to answer either Yes or No with an explanation attached, so the exact number of Yes and No answers is potentially inaccurate. There were, however, some interesting points brought up by those who answered "Other" in this way. Some of the participants acknowledged that they personally knew people who might prefer the photorealistic artwork for a variety of reasons, as everyone has different tastes. They reason that this is exactly why the concept of "westernised" artwork is fundamentally flawed; because everybody has different tastes. As such, attempting to change the marketing or style of a game to match a particular region can be considered a form of stereotyping. Some participants took particular offense to this act of stereotyping, with one participant going as far as to call it "racist".

4.4.0 – Experiment #1: The conclusion.

To summarise, the hypothesis in section 4.1.0 is wrong; the majority of the potential consumer base are not biased towards either photorealism or non-photorealism. The majority of consumers like both in equal measure, and would prefer for games of either style to receive equal representation. There is just as big of a market for non-photorealism as there is photorealism, a market which (according to the results of question **eight**) isn't being catered to.

Assuming that the mainstream game industry has a clear understanding of what the consumers want, it is evident that consumer demand/trend isn't the reason for the photorealism bias within the mainstream industry. If this is not a trend issue, however, than the issue must lie elsewhere. With the previous hypothesis proven wrong, it is now necessary to explore other potential reasons for the prevalence of photorealism.

A possible starting point for this continued research has been provided by Viktor Antonov, who suggests that the prevalence of photorealism is due to it being the easier style.

5.0.0 – Is photorealism easier?

Viktor Antonov was previously one of the designers on *Half Life 2* (Valve) and is currently the Visual Design Director at Zenimax (creators of *Dishonored*). In an interview with games™ (*Anonymous*, 2012), Antonov explained his belief that the future of computer game design lies not with photorealism, but with non-photorealism. This was supposedly the belief that had lead to *Dishonored*'s interesting visual style, combining gritty realism with a flamboyant painterly style. It is during this interview that Antonov proclaimed "anybody can do photo-realism".

Such a claim may have seemed strange a decade or so ago, when photorealism was difficult if not impossible to accomplish (as explained in section 3.2.0), but technology has since evolved to the point where photorealism is a much easier goal to achieve. With the improvement of 3D technology, many facets of the real world that artists traditionally had to replicate by hand have now become automated; the direction of light and how it bounces off particular surfaces, the shadows that are cast, the way light diffuses through skin, the way hair and cloth moves, the way the surrounding scene is reflected off a chrome surface; all of these are now easy to achieve through 3D rendering and are typically incorporated by default in the 3D suite of choice. To further understand Antonov's views, it is necessary to look at the primary differences between photorealism and non-photorealism beyond basic appearances.

5.1.0 – The elements of non-photorealism.

5.1.1 – Simplification/Abstraction.

According to an article from Computer Graphics (CG) Quarterly (*Anonymous*, 2006), non-photorealism can be quickly summarised as a simplified representation of the real world, "in which a lot of extraneous details are absent". This simplification, or abstraction, makes the imagery much easier and quicker to process for the human brain by eliminating any unimportant data.

The article continues to explain the connection between the ways a cartoon may simplify the real world, and how the human brain also simplifies the real world:

> In order to make a coherent internal picture of the physical world around us, we continually simplify the visual information coming into our eyes, picking out the essential cues needed to reconstruct what objects are around us, and our location within the scene.[…] At some point along this pipeline, our images are probably transformed into something not far from a cartoon-style representation.[…] We can think of this pipeline originating as 2D images on our retinas, and ending as [a] reconstructed 3D scene: a set of recognized objects laid out in a 3D "mental map". (*Anonymous*,2006)

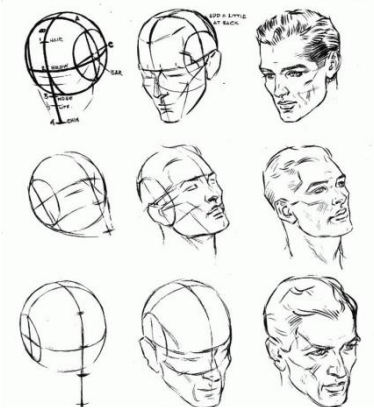


Figure 9: A step-by-step of drawing a face, starting with abstract simplification. (Loomis A., 1943)

To summarise, cartoons are a physical representation of the process the human brain goes through to process the surrounding world, breaking down what the viewer sees only into the most important components. For example, the human face can be broken into various abstract shapes *[fig 9]*. This is a common technique among artists, including those who portray realism. *Anonymous* (2006) notes this, stating that this is "exactly what visual artists have been exploring for centuries, and is at the root of many of the great artistic movements". They also note that this theory of personal perception forms the base of many schools of thought within the fine arts, and that works of fine art can be viewed as attempts by the artist to communicate the way their mind interprets the world around them; "That is to say, putting down on canvas the results from some of the intermediate stages of the visual processing that they notice taking place within their minds". (*Anonymous*, 2006)

5.1.2 – Exaggeration.

Whilst the simplification of visual information plays a large part in the creation of stylised imagery, there is another important aspect to this process; exaggeration. Many artists specialising in cartoons, once going through the process of simplification, may choose to select choice elements from the final data and bring more attention to it via exaggeration.

Within the context of character based art, this method of exaggeration is referred to as a caricature, named after the Carracci brothers "who popularized the style of overcharged or loaded portraits." (Brennan, 1982, p.5) The caricature works by exploiting the methods used by the human brain in the process of face recognition, noting which aspects of an individual's face makes it unique. For example, a person may be considered recognisable by their distinctive nose, the colour of their eyes, or their freckles. Brennan (1982, p.7) is also sure to note that this appeals to "one's mental model",

meaning that it is based on the individual's personal perception and interpretation. In this way, the concept of exaggeration is very similar to the concept of simplification; these methods are both based on how the brain interprets visual information, in this case, the defining features of a face.

For example, in this caricature of Lady Gaga *[fig 10]*, artist VampirGoth [pseud.] has chosen to exaggerate Lady Gaga's nose, chin, and eyes, to make her look more recognisable. He has also exaggerated other elements of the image, such as the size of the sunglasses, the size of her hair, as well as the shape of her elbows.

Figure 10: A caricature of Lady Gaga by VampirGoth [pseud.]. (2012)

Although Brennan's (1982) research concentrates specifically on the exaggeration of a particular individual's features, this theory can also be further expanded to cover a much wider scope. For example, a real world location can be exaggerated in the same way, taking what the brain perceives to be the most unique aspects of a location and exaggerating those features into a stylised scene. The caricature can also be exploited with purely fictional characters in order to portray a particular personality through the exaggeration of physical traits that are stereotypically associated with that personality. For example, a strong and brave character may have the size of their muscles exaggerated to emphasise their heroic status, while a young character may be drawn with significantly larger eyes to emphasise their youthful innocence.

As previously noted by Brennan (1982, p.7), this depends highly on the artist's own perception. Every artist has a different perspective on what elements of a character or scene are the most distinctive, and this personal insight will have an effect on the artwork that person produces.

For example, an artist who has a preference for conventionally cute artwork may regularly emphasise elements that make a character appear cute, which may involve making their eyes larger, or making other features smaller and more delicate. Conversely, a fetish or pin-up artist may exaggerate characteristics that they find sexually appealing, whether this may be individual body parts (such as the breasts) or the entire physique. This alone already provides many different ways of presenting the same character and/or scene.

5.1.3 – Photorealism in comparison.

Taking these points into consideration, there is stark difference between photorealism and non-photorealism. While photorealism has only one appearance, non-photorealism has an infinite amount of possible appearances, all depending on the way the artist interprets the world around them. This is the reason why Antonov claims photorealism is easier, further reasoning that anybody can "scan things into a game" (*Anonymous*, 2012). As photorealism doesn't require the aforementioned processes of simplification and exaggeration, elements of the real world can be translated directly into a videogame through the use of photography (textures) and 3D scanning (meshes), thereby making the process both easier and faster. This is particularly true with games like *Half Life 2* or *Heavy Rain*, both of which feature characters whose appearances are directly based off real people with little to no alterations, the latter of which makes heavy use of 3D scans.

With photorealism, the real world can be directly translated into the video game world with little to no alterations. With non-photorealism however, it needs to go through the additional steps of abstraction and exaggeration (both of which can be affected by the individual artist) before finally being used in the game. It is a comparison between a very technical way of viewing world, and a very abstract and personal way of viewing the world.

While this does help to rationalise Antonov's views and help establish photorealism as the easier of the two methods, it also opens up the possibility that games designers are naturally predisposed to the use of photorealism.

5.2.0 – The default bias .

5.2.1 – A technical art form.

As established in section 5.1.3, photorealism is a technical way of viewing the world, while non-photorealism is an abstract way of viewing the world. With this established it can be theorised that computer generated imagery (CGI) in general has a natural predisposition towards the use of photorealism.

The reason for this is due to the creation of CGI being in itself a highly technical process. While traditional artwork might utilise mathematical rules such as the golden ratio in the creation of their works, these rules are only used as a reference point and not as the tools themselves. CGI however, being a digital art form, is powered purely through the use of mathematical equations.

For example, there is a large difference in the way colour is viewed in both traditional and CGI art. In traditional art, colours are simply that. Primary colours can be mixed to produce secondary colours, and secondary colours to produce tertiary colours. In CGI art however, colours are numbers, and the way these numbers are used will affect the colour being displayed. These numbers include RGB (Red, Green, Blue) values, HSV (Hue, Saturation, Value), CMYK (Cyan, Magenta, Yellow, Black), and Hexadecimal codes (e.g Black is #000000). Just like any other number, these can be added, subtracted, multiplied, and divided to change the colours being shown.

As another example, a basic sphere. While a traditional artist may simply draw a sphere, whether it be freehand or with a compass, a 3D CGI artist will instead have to generate a 3D sphere. This 3D sphere consists of a series of vertices that have been placed in 3D space, each with a set of numbers dictating where these vertices lay on the X, Y, and Z axes. These vertices then need to be connected together in a specific pattern in order to create polygons, forming the surface of the sphere.

$$k_{\text{spec}} = \frac{1}{\sqrt{(N \cdot L)(N \cdot R)}} \frac{N \cdot L}{4\pi\alpha_x\alpha_y} \exp\left[-2\frac{\left(\frac{H \cdot X}{\alpha_x}\right)^2 + \left(\frac{H \cdot Y}{\alpha_y}\right)^2}{1 + (H \cdot N)}\right]$$

Figure 11: The mathematical equation for anisotropic specular. (Ward anisotropic distribution, n.d.)

While modern 3D modelling software allows the user to create such shapes with ease, automating most of the process, there is still a staggering amount of data that the end user has to contend with in order to create a final product. This data is labelled using specific technical jargon, including terms such as diffuse, raytrace, ambient occlusion, global illumination, subsurface-scattering, lambertian

reflectance, anisotropic specular *[fig 11]*, surface normals, and weight-mapping, to name a few. This language is vastly different to the language utilised by traditional artists, including terms such as tones, shape, form, shade, tactile lines, strokes, and hatching. (Carter, 2011, pp.11-13)

From this it can be established that the art of creating a 3D image is a process which requires not only a certain level of artistic skill, but also a certain level of technical understanding. One of the main reasons for this is due to the fact that the evolution of 3D imagery has taken cues from the only other 3D reference available; the real world.

5.2.2 – The science of what we see.

The real world in itself is very technical. Every visual aspect of the world that we see can be explained through scientific and mathematical reasoning. For example, the way that light reacts to particular surfaces in the real world can be explained by the laws established by Johann Heinrick Lambert in his book *Photometria*, published in 1760. These included 'Lambert's law of absorption', which describes the exponential decrease of light as it is absorbed by a volume of uniform transparency (such as water), and 'Lambert's cosine law', which dictates the appearance of light depending on the angle of the surface in relation to the viewer. (O'Connor and Robertson, n.d.) It is this same technical way of looking at the real world that has powered the evolution of 3D imagery, with such equations and theories being used to calculate the data needed to correctly render light, shadows, and highlights *[fig 11]* among other effects. Lambert's laws themselves have been integrated into 3D imagery, with one of the most common shading models being named "Lambert" after the cosine law it is based on. It is this common source of language that makes photorealism, the recreation of the real world, fit neatly into 3D computer generated imagery with no issue.

Non-photorealism on the other hand isn't traditionally a technical effort. It is a personal and abstract effort. As established earlier in section 5.1.0, non-photorealism relies heavily on an artist's individual interpretation of the world around them, as opposed to recreating the original source with pin-point accuracy. The art of non-photorealism requires input predominantly from the human mind for the purpose of abstraction and exaggeration.

A computer, while intelligent enough to compute a large amount of data in little time, is not a replacement for the human mind; it is not capable of interpreting and subsequently abstracting the world the same way a human would, at least not without a large degree of input from the user (in this case, the artist). In order to provide this necessary input, the artist must have a clear understanding of both the way the tool works and the language used in order to overcome the technical limitations of the program. This is difficult to do, as the programs used to create 3D graphics were clearly not created for traditional artists, but specifically for 3D artists.

5.2.3 – The default bias: The concept.

This is an issue observed by Rautek, Bruckner, & Gröller (2008) when observing the issues faced by traditional illustrators when using 3D rendering software for scientific visualisation. While Rautek, et al. acknowledges that over the years these tools have evolved to allow illustrators to transition into the realm of 3D rendering much easier than before, these illustrators still aren't using the software to its full potential, i.e. to create expressive imagery. Rautek, et al. stipulates:

> One problem is that visualization software is frequently designed specifically for domain experts, such as research scientists or engineers, and utmost care is taken to accurately present the original data. An illustration, on the other hand, deliberately simplifies or distorts the data according to its communicative intent. This difference in focus often leaves too little flexibility for the illustrator. Moreover, user interfaces employ the language of the domain experts while illustrators are more used to terms and concepts used in graphical arts. (Rautek, et al., 2008)

While this quote specifically discusses this issue in the context of scientific visualisation, the same issue presides in the 3D CGI industry. In this context, the domain experts are the 3D artists, while the users being affected are the traditional artists. While visualization software is designed to accurately present the original research data, 3D suites are designed to accurately render lighting and shadows based on real world laws (as established in section 5.2.2). Furthermore, the issue of the user interface employing language that only the domain experts understand is an issue shared by artists in the 3D CGI industry, as established in section 5.2.1. The unintended side-effect of these language and technical barriers is a system that favours photorealism over non-photorealism; a default bias in the tools themselves that limit the traditional artist's ability to fully realise their vision.

5.2.4 – The default bias: Examples.

With this concept now established, it is easier to understand why there is such a stark difference between conceptual and final art in many game projects. In many of these cases it would actually be more difficult (if not impossible) to accurately portray the style in 3D, making photorealism the easier and more likely option. This issue can be seen in action in the following examples.

Figure 12: Hatsune Miku's official design. (Kei, 2007)

5.2.4a – Project Diva: Hatsune Miku

First is the *Project Diva* series of games produced by SEGA, featuring the Vocaloid character Hatsune Miku. Figure 12 shows the original design of Hatsune Miku as she is portrayed in official promotional material, including the original box art for the *Vocaloid* software of which she originates (designed by Kei Garō). The artwork is shaded in a manner not too dissimilar to watercolours, with soft shades of cool colours, and in some places a splotchy appearance. Both her clothes and her hair have a noticeable metallic quality to them, rendered in such a way to suggest the reflection of a surrounding scene. The hair in particular features pronounced highlights that are common to anime images.

Compare this to her appearance in the original PSP game. Although stylised in appearance, the style itself is vastly different to that of the original design. The colours are more vibrant, and the shading has been simplified. The clothing and hair have lost their metallic quality. Furthermore, the outlines present in the original design are no longer present. *[fig 13, left]*

Figure 13: (Left) *Hatsune Miku: Project Diva*. [PSP] (Jenni [pseud.], 2009)
(Right) *Hatsune Miku: Project Diva - Dreamy Theater*. [PS3] (Spencer [pseud.], 2010)

It should be noted however that these changes were potentially due to the limitations of the hardware rather than the tools used. While the PSP was the most graphically advanced handheld at the time of its release, it still has much lower specifications when compared to a home console or a PC, capable only of rendering simplistic shaders. As such, not only is the PSP incapable of rendering such a complex non-photorealistic style, it's also incapable of rendering in a photorealistic style. *Project Diva* takes this further by not using shaders at all, instead relying on simplistic shadowing painted directly onto the textures in order to give the character visible depth.

Since then however, *Project Diva* has been released on the Playstation 3 through the *Dreamy Theater* expansion pack. *[fig 13, right]* This pack allows the player to play *Project Diva* in HD using the Playstation 3, providing superior visual quality through the use of high definition models as well as the introduction of shaders. The Playstation 3 is a powerful home console that is capable of outputting visually detailed titles such as *Heavy Rain*, *Enslaved*, and *Final Fantasy XIII* due to its higher specifications, capable of utilising higher resolution models, textures, and complex shaders. In spite of this however, rather than use this additional power to try and more accurately recreate the painterly style of the original design, SEGA instead used this additional power to add realistic shadowing, normal mapping, and metallic shaders on what is still an anime styled character. This gives Hatsune Miku the appearance similar to that of a resin figurine, a style that leans much farther away from its painterly roots than before, and much closer towards photorealism.

Taking into consideration the aforementioned (section 5.2.0) possibility of photorealism being a much easier and more efficient style to use, it can be theorised that this was the reason for this drastic style change.

5.2.4b – Fable 2

The same discrepancy can be observed in the role-playing game *Fable 2*. The original concept artwork for *Fable 2* portrayed the world and inhabitants of Albion in a very dirty, gritty style. *[fig 13]* The art provides a very traditional look to them, featuring predominant use of pencils and watercolours in such a fashion that they give the image a very distinct texture; the pencilled cross-hatching provides depth and detail, while the paints produce a natural cloudy texture. The somewhat erratic use of paint and pencil strokes combined with the desaturated colour scheme give

the impression of a grim and dirty atmosphere, offset by the expressiveness and flamboyancy of the designs themselves.

Figure 14: *Fable 2* conceptual art. (Fable 2, n.d.)

The final game however forgoes these details in favour of a much cleaner look, one that is comparable to the works of Pixar. Although the designs of the characters are still stylised and expressive, the game now utilises realistic textures, realistic lighting and shadowing, as well as realistic shaders for materials such as metal, cloth, hair, and skin. The erratic sketching and painting is no longer present, with the game favouring a much cleaner, softer look. *[fig 14]*

Figure 15: *Fable 2* in-game screenshot. [Xbox 360] (Hofer B., 2009)

Taking the possibility of photorealism being a more efficient style into consideration again, this visual difference could be justified by these shaders being much simpler to produce and use.

6.0.0 – The Default Bias: Proving the theory.

As established in section 5.2.0, there is a potential bias towards photorealism within the design of the tools used, utilising features and language that favour photorealism while making the application of other art styles more difficult and time consuming, complications which in themselves can be detrimental to the game production process. This suitably matches Viktor Antonov's claims of

photorealism being the easier style to utilise (section 5.0.0), while at the same time providing a possible explanation for the drastic changes made between concept design and final product in particular projects (section 5.2.4). In its current form however, the default bias is only a theory.

6.1.0 – The Default Bias: The experiment.

In order to prove (or disprove) the existence of the default bias within game production tools, it is necessary to experiment with the tools in question. As such, an experiment will be carried out using the Unreal Development Kit (UDK), a popular game engine produced by Epic Games. This is also the same engine used to produce *Samaritan*, the photorealistic tech demo described in section 3.1.0. On the other hand, the Unreal Engine was also used to produce the *Borderlands* games (Gearbox), a series of first person shooters with a simplistic comic book aesthetic. UDK has been chosen for this experiment for multiple reasons; the engine is freely available, it is notable for producing photorealism, and it is widely used within both the independent community and the mainstream industry. Some examples of previous games made with the Unreal Engine include *Mirrors Edge* (DICE), *Duke Nukem Forever* (Gearbox), *Dishonored* (Bethesda), *Mass Effect 3* (BioWare), *Super Monday Night Combat* (Uber Entertainment), *Batman: Arkham City* (Rocksteady Studios), M*ortal Kombat* (NetherRealm Studios),

This engine has previously proven itself capable of producing photorealism and non-photorealism to an extent. The latter however is the minority of the two, and in most cases where non-photorealism is used it usually amounts to the same simplistic toon styles described by Doug Cooper in section 3.4.2. For this experiment, its ability to produce more complex and varied forms of non-photorealism will be judged.

This will be done through three separate experiments. Each experiment will use a different art style as its reference point. These art styles include; a somewhat simplistic visual novel style, a manga screen tone style, and a detailed painterly style. For each experiment, the art style used will be analysed and deconstructed into its basic points using the language of both the traditional artists and the domain experts. UDK will then be used to create the closest approximation of the style possible, primarily through the creation of shaders, with each step of the process being documented in detail.

6.2.0 – The Default Bias: Hypothesis.

As noted in the previous section, UDK is notable for producing photorealistic results. As such, it is highly possible that the engine will suffer from the default bias in some way. This forms the first part of the hypothesis; the theory that the default bias is true, and it is present in UDK. It is unlikely, however, that the default bias will manifest itself when producing very simplistic imagery. As such, the second part of the hypothesis is that the default bias will only manifest itself during the later, more visually complex experiments; the more complex the style, the higher the possibility of the default bias manifesting itself as an obstacle.

7.0.0 – Experiment #2: visual novel style shaders.

For the first of these three experiments, a relatively simplistic toon style will be recreated in the form of a real-time 3D shader. Specifically the toon style chosen is that of a typical Japanese Visual Novel. A Visual Novel (commonly referred to as a Dating Sim) is a simplistic game, usually romance

themed, that relies predominantly on 2D artwork and text to convey a story. Most Visual Novels play like a choose-your-own-adventure (CYOA) book, providing the player with only limited interaction through occasional multiple-choice decisions, however this isn't necessary; *Higurashi no Naku Koro ni* (a.k.a *Higurashi When They Cry*) by 07th Expansion is a notable example of a linear Visual Novel with no branching paths.

The artwork featured in a typical Visual Novel shares much of the simplicity of anime/toon artwork (such as cel-shading). Due to the nature of Visual Novels, however, the artwork becomes a lot more visually detailed; the lack of both animation and unique images allows the artist to spend more time on the artwork, both drawing in more details as well as using more detailed shading.

Figure 16: Suzuna Kuraki from *Kao no Nai Tsuki*. (Carnelian [pseud.], 2000)

For the purpose of this experiment, figure 16 will be used as the source material. Figure 16 is a picture of the character Suzuna Muraki from the Visual Novel *Kao no Nai Tsuki* (a.k.a *Moonlight Lady*), drawn by manga artist Carnelian [pseud.].

7.1.0 – Dissecting the style.

7.1.1 – Shading and Tones / Diffuse.

The way that a surface has been shaded, as well as the tones used, can have a dramatic effect on how the surface looks. Choosing the right combination of shading and tones can produce anything from a basic toon drawing, to a realistic portrayal of skin, velvet, or brass. In 3D rendering, the way a surface has been shaded is referred to as the diffuse. A typical diffuse shader will start with the base colour at its brightest point, and end with black at its darkest point, with an even gradient between the two points. In the language of a traditional artist, this may be known simply as shading, and the colours used may be referred to as tones.

This character, however, clearly does not use the typical diffuse shading as described before. Black is never used; instead a slightly darker and more vibrant derivative of the base colour is used. Amongst toon artists, this is known as duo-tone shading, literally referring to the use of two tones (the base, and the shadow). Some of the other surfaces, such as the hair and clothing, can use up to three

colours; a base colour, a mid tone, and the darkest tone (known as tri-tone shading). Furthermore, instead of an even blend between these tones, these tones instead use much sharper transitions. This is reminiscent of a style commonly known amongst traditional artists as cel-shading.

Cel-shading as a term has its origins in traditional cel-based animation, and refers to the time saving techniques used for traditional animations in order to produce the needed individual frames of animation much more efficiently. Traditionally, animation frames were individually drawn onto layers of acetate, with the necessary colours painted on the back behind the outlines. Due to both the time it takes to produce a single frame of animation as well as the tools being used, the colouring process had to be simplified in order to accommodate the simplification of shadows into their most basic shapes. The term cel-shading is literally referring to the style of shading used in the creation of animation cels. Although animation tools have since advanced to allow much more detailed shading and animation, many shows still resort to cel-shading both for its efficiency and its charm.

This image is not a part of an animation however, and as such the artist can afford to have slightly softer and more detailed shadows in the image. Using cel-shading as a starting point, he has applied a soft blur to many of the more rounded surfaces (such as her hands) while using much sharper shading for any shadows cast by objects that are close to the surface (such as the hair casting a sharp shadow onto her forehead), similar to how shadows work in the real world; the closer an object is to a surface, the sharper the shadow it casts.

Returning to the domain language, the artist would need to take your basic diffuse, replace black as the darkest point with a colour of their own choice, and adjust the transition between the lightest and darkest points to be much sharper (though still with a soft blur).

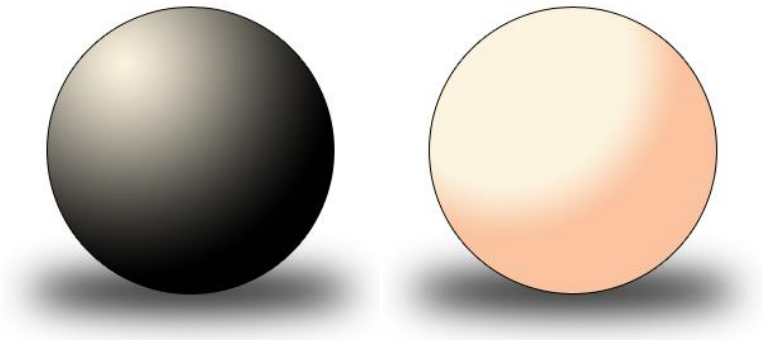


Figure 17: (Left) A representation of a normal diffuse with a flesh toned base colour. (Right) A representation of the diffuse shading used in the source image on the skin.

7.1.2 – Tone Variation / Ambient Occlusion.

The shading appears more detailed than that however, particularly in the colours or tones used. Upon closer inspection, there are various areas which utilise soft gradients of colour that help to make the shading appear much deeper and more detailed. This can be observed on her hands (particularly the creases her fingers make when bent), her neck, her left cheek, and her forehead. Rather than being caused by shadows cast by the primary light, these appear to be caused more by surface proximity. Her bangs are very close to her forehead, and a gradient is used here to represent this. There is a clump of hair near her left cheek, and a gradient is used to represent this (the shadow

could not have been caused by the primary light as it is in the wrong direction and isn't sharp enough). Her neck is predominantly obscured by her chin and her clothing, and a darker colour is used to represent this.

These points of interest suggest the use of ambient occlusion, a method of shading a model based on light attenuation caused by occlusion.

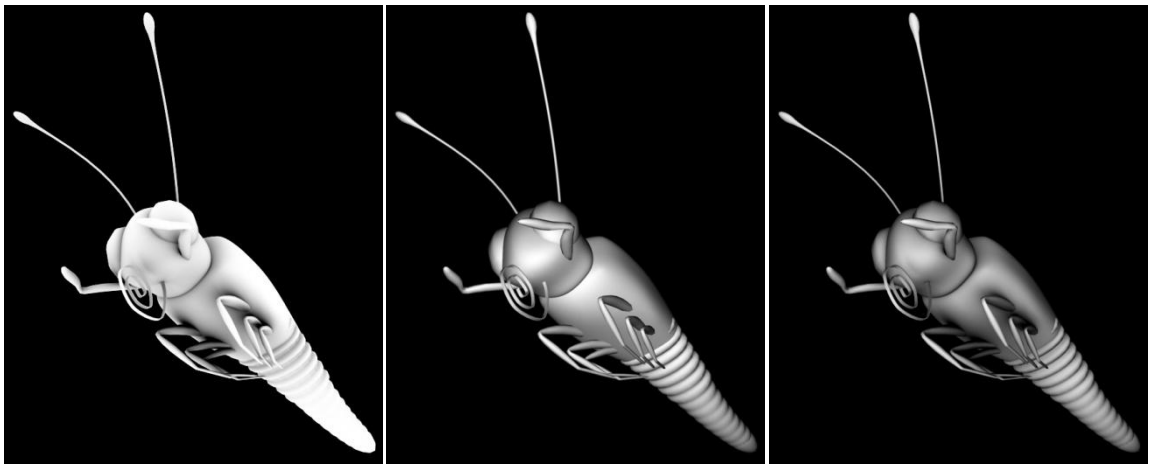


Figure 18: (Left to Right) Ambient occlusion only, diffuse only, and the two processes combined. (Mrtheplague [pseud.], 2005a; b; c)

As can be observed in the above examples, ambient occlusion produces soft shadows in any crevice-like area, particularly in areas where two shapes connect together (such as the legs). This shading helps to provide much more depth to the image, while also providing a much stronger effect of surfaces being close or connected to each other. Much like diffuse shading, ambient occlusion shading starts from white at its lightest point, and black at its darkest points, with a mostly smooth gradient between the two points. The ambient occlusion can be utilised to achieve the deep shading required by the style.

To do this, the artist will need to replace black as the darkest point with a colour of their own choice. Upon closer inspection, it appears as though the gradient consists of two colours; an orange tone for parts of the surfaces that are lit, and a much more pink tone in areas that are in shadow. This shows that the colour of the ambient occlusion is directly influenced by the colours used in both the diffuse texture and shading, possibly through multiplication.

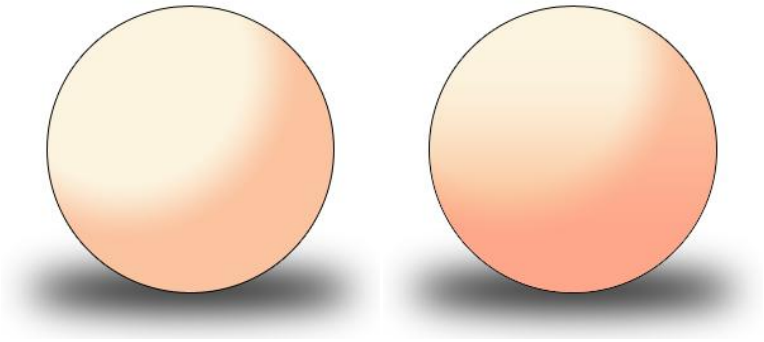


Figure 19: (Left) A representation of the diffuse skin shading without ambient occlusion. (Right) A representation of the diffuse skin shading with ambient occlusion.

7.1.3 – Outlines.

Like most toon images, this style features prominent toon outlines. Unlike most anime styled animations however, which commonly use even width outlines, this image instead uses outlines that vary in width, giving the effect of outlines produced using traditional inking methods. It is common practise when inking outlines in such a manner to vary the line width based on the lighting of the surface (lit surfaces with thinner lines, shadowed surfaces with thicker lines), however this doesn't appear to be the case in this particular style. Instead, outlines vary in width depending on how straight or curved the particular edge is. This is typically caused by straight lines being inked with a simple quick movement (resulting in thinner lines), while corners and curved surfaces are inked with slower and more precise movements (resulting in thicker lines). The colour of the outline itself also varies depending on the lighting of the surface.

In 3D rendering, this kind of outline can be produced by using fresnel (also referred to as edge blend in some 3D packages), a function which determines the boundaries of an object based on edge angles. For example, if a camera is pointing directly at a sphere, the polygon in the centre of the sphere is facing the camera almost head on. According to the fresnel function, this is the centre of the mesh, and is represented as white. Towards the edges of the sphere, the polygons start to face away from the camera to the point where they are perpendicular to the camera. According to the fresnel, this is the edge of the object, and is represented as black. Any polygon facing in the opposite direction of the camera is not factored into the equation. The fresnel would then typically blend between the centre point and the edge point with a smooth gradient. Because this is based on the angles of an edge, rounder, more concave objects tend to have thicker outlines.

To produce an outline for the shader, information provided by the fresnel should be taken and sharpened in a similar fashion to the diffuse. Information from the diffuse would also need to be utilised in order to alter the colour of the outline.

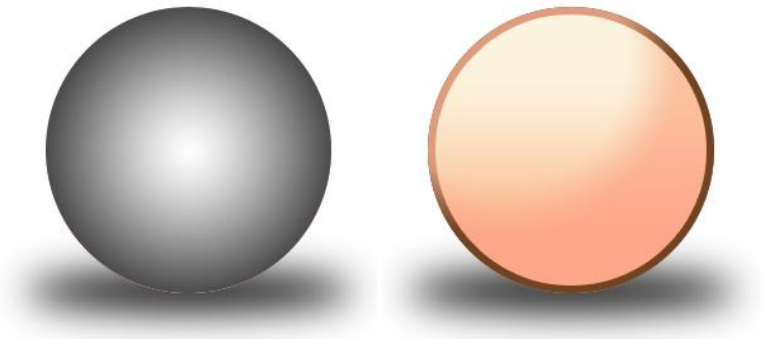


Figure 20: (Left) A representation of the default fresnel.
(Right) A representation of a sharpened fresnel, with diffuse based colouring.

7.1.4 – Highlights / Specular.

Figure 21: Shiny hair in the real world. (Flickr flame [pseud.], 2009)

Specular refers to the highlights that may appear on particular surfaces, especially ones that are shiny. In this case, the most noticeable highlight in the image is on the hair; a thin, horizontal stripe that gives the impression of healthy, silky hair. This is in fact an exaggeration based on reality; hair in the real world exhibits similar horizontal highlights in well lit areas *[fig 21]*. This specific highlight is known as an anisotropic specular, a specular highlight used for surfaces such as hair and brushed metal. This highlight however is stylistically different to real world specular; the highlight begins as a sharp, white stripe, subsequently fading into a soft purple haze. This makes the overall image appear much brighter and more vibrant. To produce this effect, an anisotropic specular will need to be altered in a very similar fashion to the diffuse, customising the shape and colour of the highlight.

7.2.0 – Replicating the style in UDK.

This section of the document explains in detail the process of translating this style into a shader that can be used in real-time 3D. For the sake of comparison, a 3D model was produced resembling the previously shown character (section 6.1.0).

7.2.1 – Diffuse (lambert).

As stated before in section 6.1.1, it is necessary to change the basic diffuse shading into something sharper and more colourful. By default, UDK gives users the option of various hardcoded lighting modes, including phong, blinn, anisotropic, and subsurface scattering, all intended to produce realistic effects. In order to alter the diffuse the way that the user wishes, it is necessary to open access to the lighting information, which requires the use of a custom lighting mode.

The custom lighting mode is a mode with no hardcoded shading such as phong or anisotropic, allowing you to instead build your own lighting from scratch. As such, the first step is to create basic lambertian shading which can then be used as a starting point for the rest of the shader. This is done by taking the surface normal (typically in the form of a normal map), the light vector (the direction of light), and output a dot product of the two. This will produce the basic diffuse shading; white where the light is brightest, black where the light is darkest. Note that here, an additional clamp node was used (values: 0.01 and 0.99) to fix a minor glitch caused by UDK.

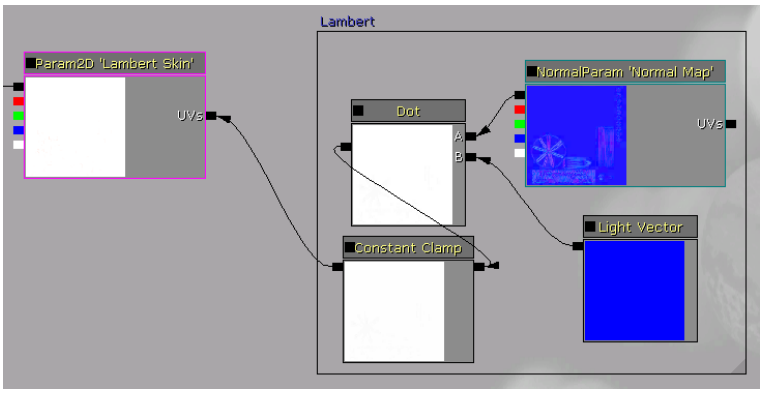


Figure 22: A lambert group plugged into a gradient ramp.

Now that access to this shading data has been opened, it is possible to alter it to appear cel-shaded. The most efficient method to do this is with a gradient ramp.

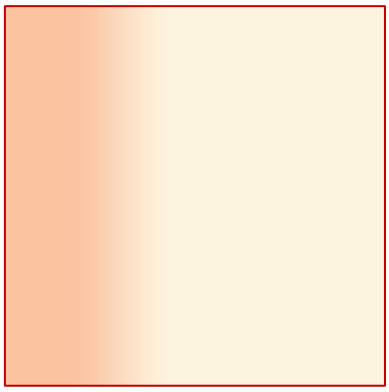


Figure 23: A gradient ramp (skintone).

A gradient ramp is a texture consisting of horizontal gradient; left side being the brightest, and right side being the darkest. Typically, a texture will use a mesh's own mapping as a UV map. In this case however, the lambert shading is used as the UV map. This causes the texture to dynamically wrap around the mesh based on the light direction. By using a gradient map with bright colours and a sharp transition, the effect of cel-shading is achieved.

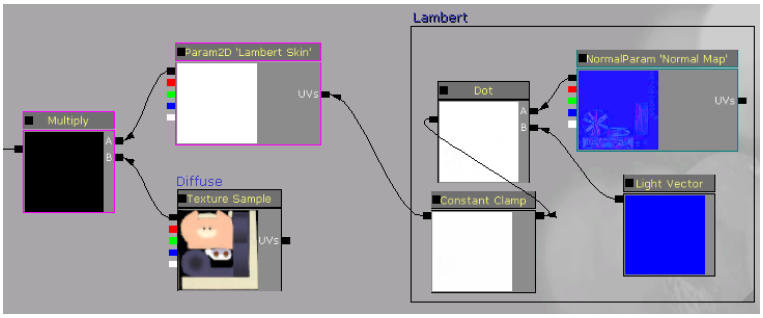


Figure 24: A diffuse based gradient ramp multiplied with a texture.

This effect is then multiplied over the diffuse texture by using a multiply node. This produces a textured model with ramped shading.

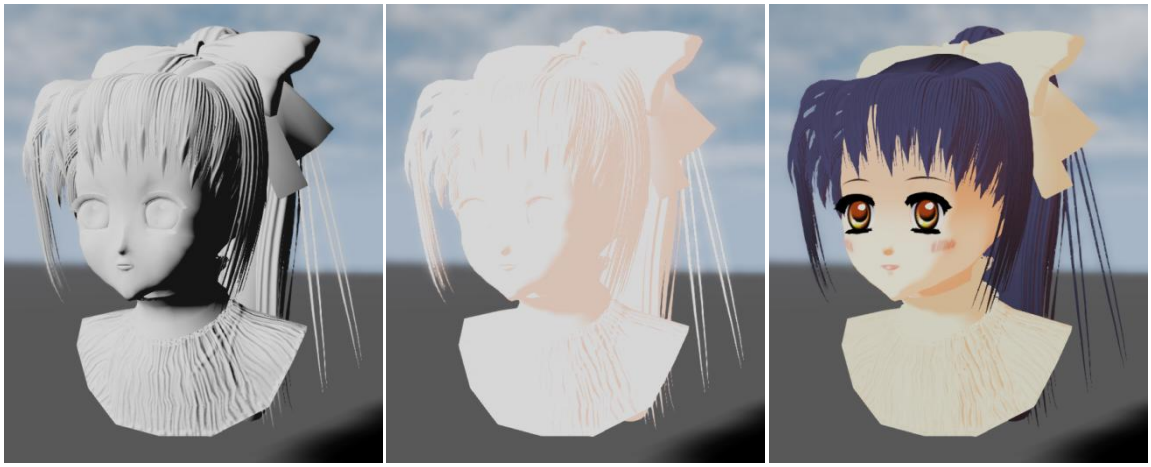


Figure 25: (Left to Right) Base diffuse, gradient ramp, and final diffuse shading.

7.2.2 – Outlines.

There are three ways to achieve a toon outline in real-time 3D:

1. Duplicate the mesh, expand it, and reverse the normals. This produces a secondary mesh that appears to be turned inside out. As the base mesh blocks most of the secondary mesh, this produces the effect of an even width outline. This is a very basic method that has been used in early 3D cel shaded games such as *Sheep-dog n' Wolf* and *Jet Set Radio*. On the one hand, the outline is of an even width, and can be rendered in any colour that is desired. Conversely, however, this doubles the polycount of a single model, which isn't appropriate for more fluid/hand drawn effects, and also doesn't work with meshes that rely on 2-sided transparency (for example, characters featuring layered hair) as the secondary mesh would also be 2-sided.

2. The second method utilises a post-process shader. This typically works by taking the z-depth information of an entire scene and then utilising this as the source data for sobel edge detection. As this is a shader rather than a mesh, it can be further expanded to change the style of the outlines, such as making it uneven, having outlines become thinner and thicker depending on depth, or changing the tolerance of the outline (making it either more simplified or more detailed). Conversely, however, the colour of the outline cannot be customised for individual surfaces in a scene, instead only being customisable for the entire scene. While this would work for styles that only require outlines of a single colour (such as a basic comic book style), it doesn't work for more complex styles which use coloured outlines (such as Disney animation, or the Visual Novel style this experiment is dedicated to).

3. The third and final method is to use the aforementioned fresnel function (section 6.1.3). The fresnel function is a pre-produced dot product output of both the surface normal and the camera vector, specifying which polygons are facing the furthest away from the camera (and as such, presumably the edge of the mesh). On the one hand, the colour/thickness/sharpness of this outline can be customised for each individual surface. Unfortunately though, it is impossible to get an even width outline using this method, so it is more appropriate for styles that have a more fluid/hand drawn look to them. For example, the outlines in *Street Fighter IV* are generated using this method to match the heavy ink style of the concept art.

The fresnel method appears to be the most suitable method for this experiment. To do this, the user must first begin with a fresnel node, producing a black to white soft edge effect. In order to produce a much sharper result, a gradient map is used again; the left side of the ramp is the edge, and the right side is the centre. This produces a solid black edge.

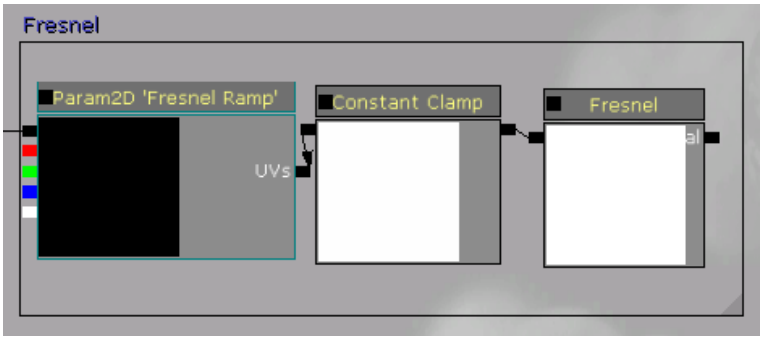


Figure 26: A fresnel plugged into a sharp Gradient Ramp.

Figure 27: The outline Gradient Ramp (outlined for clarity).

Now it is necessary to combine this outline with the diffuse shading. This is done using linear interpolate (also referred to as lerp). This node allows for the blending between two different lerp inputs via an alpha input. Here, the sharpened fresnel will be used as the alpha input, setting the established diffuse network as the second lerp input.

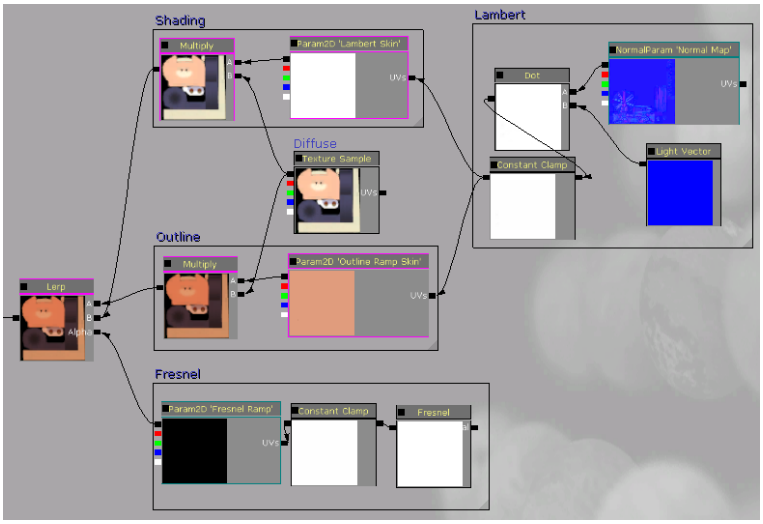


Figure 28: Applying colour to the outline using lerp.

The other input can be used to colour the outline. A constant3 could be used to produce a solid colour, but this style requires an outline that changes colour depending on the lighting. To do this, a second lambert based gradient ramp (darker than the base colour) is used as the first lerp input. This produces an outline which changes colours depending on how the scene is lit.

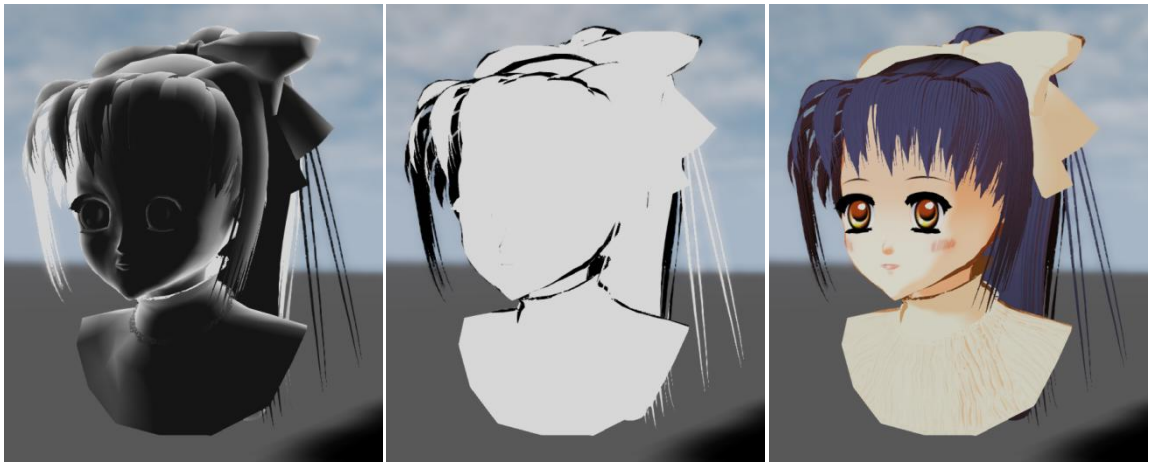


Figure 29: (Left to Right) Base fresnel, fresnel outline, and final outlined surface.

7.2.3 – Ambient occlusion.

This is the point where the first concession must be made. While software renderers allow the user to set ambient occlusion on a surface by surface basis, this is impossible to do in a real-time environment. Surface based ambient occlusion requires too much memory in order to be suitable in real-time 3D. Instead, UDK utilises Screen Space Ambient Occlusion (SSAO), a post-process shader which produces a rough calculation based on the scene normals. Not only does this tend to produce a slightly noticeable halo around most objects, but it is also not possible to change the colour of the ambient occlusion on a surface by surface basis. Due to this, the ambient occlusion shader cannot be utilised in the intended way for this style, and as such it is necessary to fake the effect instead.

The optimum and most common way to fake ambient occlusion is through the use of textures. This is typically done through the generation of an ambient occlusion map. Depending on the type of model, this can be done either by baking ambient occlusion shadows into a texture via a 3D suite such as 3DS Max (better for more detailed objects), or by hand painting them (better for much simpler or stylised objects). After that, this map can be utilised either on its own (as part of a shader), or combined with the diffuse map into one texture (saving memory).

In this case, the ambient occlusion has been combined with the diffuse texture, producing the necessary effect while being mindful of system memory.

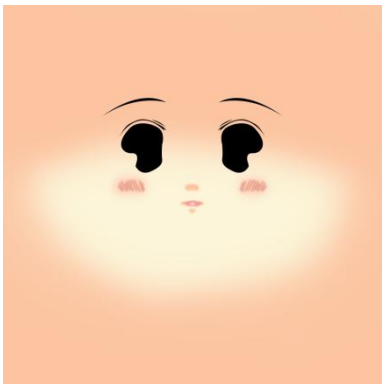


Figure 30: A close-up of the face texture, with simple hand-painted ambient occlusion.

7.2.4 – Specular.

As previously established in section 7.1.4, the only noticeable specular highlights in the source image are the highlights on her hair, a representation of anisotropic specular. While UDK does support the use of anisotropic highlights, they are included only in the form of a hardcoded lighting mode. As such, the built in anisotropic specular cannot be used along with the custom lighting mode that is currently used, and must be manually rebuilt.

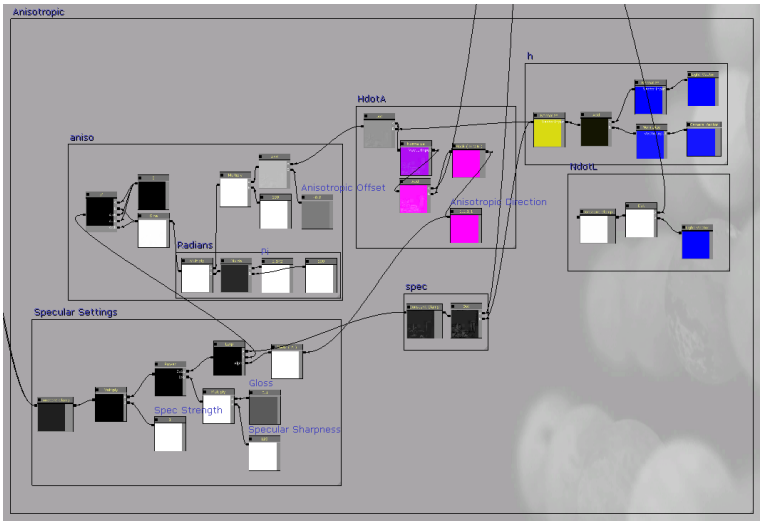


Figure 31: Complete anisotropic specular shader network.

The above node network is based on the *Anisotropic Highlight Shader* developed by O'Hare (2012), originally for use with the *Unity* engine. Using the original code as a reference, the anisotropic specular effect has been recreated within UDK, a feat made possible by both tools utilising the same technical language.

For example, the below piece of code:

fixed3 h = normalize(normalize(lightDir) + normalize(viewDir));

Can easily be visualized with UDK's material editor as shown in figure 32.

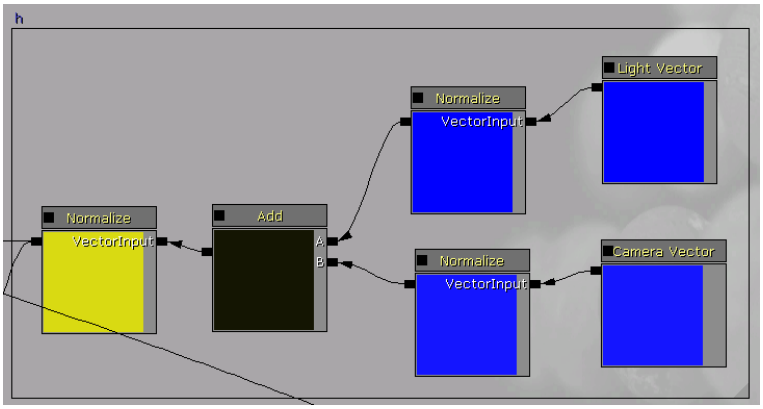


Figure 32: The "*fixed3 h*" shader network.

7.2.5 – Specular Issues.

Some issues were encountered within this process however, as there are some segments in the code that do not have UDK equivalents, and as such had to be rebuilt from the ground up. Both of these cases can be viewed in the below line of code:

*float aniso = max(0, sin(radians((HdotA + _AnisoOffset) * 180)));*

7.2.5a – Radians.

The first was the radians method, used to convert degrees into radians (an alternative way to measure angles). While traditional coding has a radians method built in, UDK does not have a radians function within the material editor. Instead, the degrees must be manually converted into radians using the formula *"angle in radians = angle in degrees * Pi/180" (Dendane, 2012).*

As UDK also does not have a built in Pi function in the material editor, a constant with the value 3.1415926535897932 was used in its place. *[fig 33]*

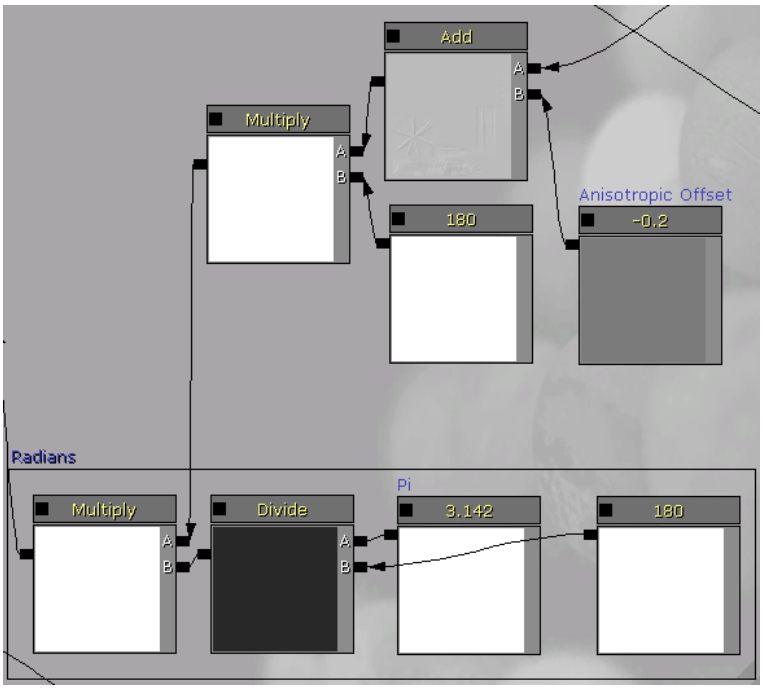


Figure 33: The radians shader network

7.2.5b – Max.

The second method is the max method, used to compare two values and output the largest. In its place, an if node was used instead. An if node has five inputs; *input A*, *input B*, *if A > B*, *if A = B*, and *if A < B*. The if node was set up to use *input A* in any case where *A* is greater than *B*, and *input B* in any case where *A* is smaller than *B*. *Input B* is a sine pattern generated from the previously computed radians. This sine pattern ensures a smooth blend between 1 (brightest) and 0 (darkest), but can also go past 0 into the negatives, thereby creating an unwanted busy pattern. This if method (used in place of the max method) is a non-destructive way of ensuring the pattern does not dip below 0.

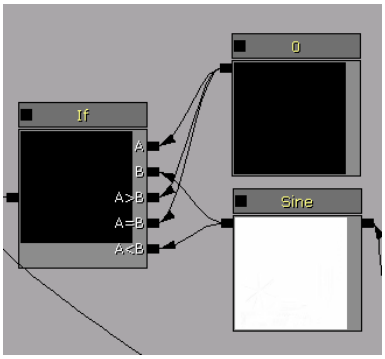


Figure 34: The max shader network.

With these changes in place, it is then a simple matter of altering some of the values (gloss, sharpness, strength, direction, and offset) to the necessary settings.

Once the anisotropic specular is created and ready for use, it goes through two separate processes. The first process is a linear interpolate. The first lerp input is set to the diffuse shading, and the second lerp input is set to a light purple. This produces a purple specular that smoothly blends into the base texture. *[fig 36, upper right]*

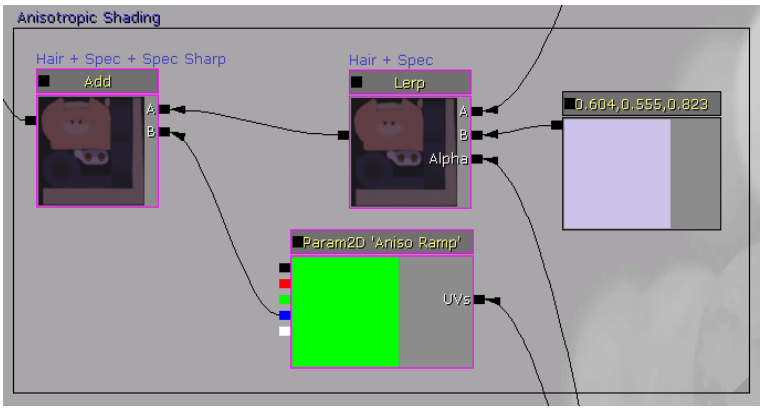


Figure 35: Incorporating the anisotropic specular to the rest of the shader network.

The second process is gradient ramp. This produces a thin, sharp, white highlight. *[fig 36, lower left]* These two processes are then added together to create an anisotropic specular effect which, much like in the source material, has a sharp white highlight followed by a light purple haze.

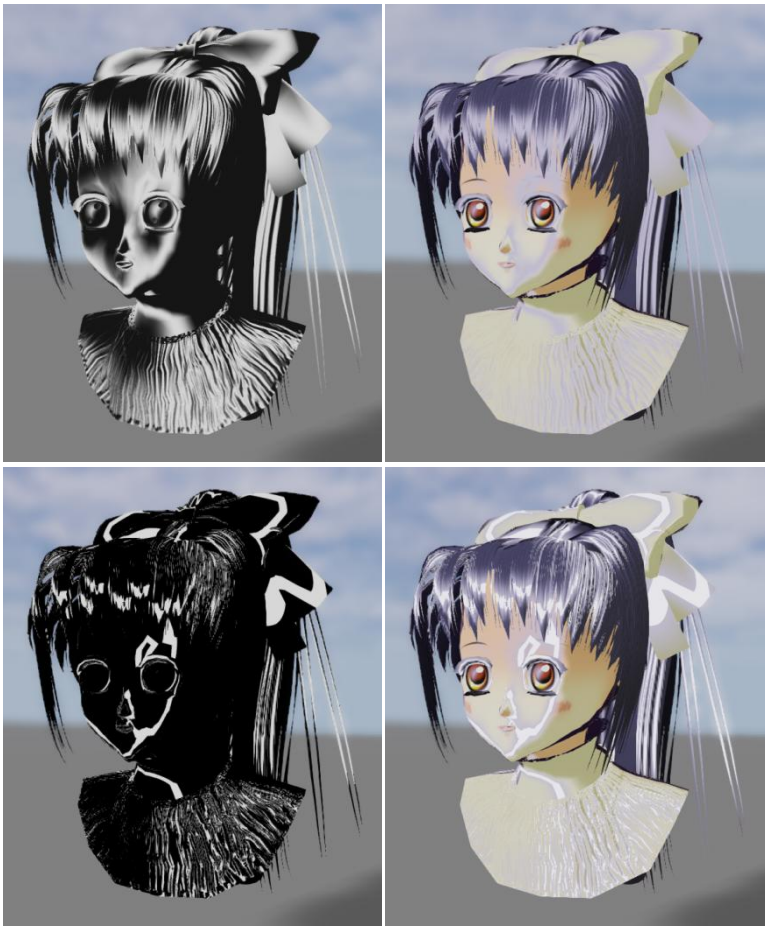


Figure 36: (Upper Left to Lower Right) The base anisotropic specular, the purple haze process, the sharp white process, the final anisotropic specular effect.

7.2.6 – Masking.

As the style requires the use of different coloured outlines and shading on different parts of the model (including the anisotropic specular, which should only be used on the hair), it is necessary to isolate different shader settings to different sections of the model via a mask.

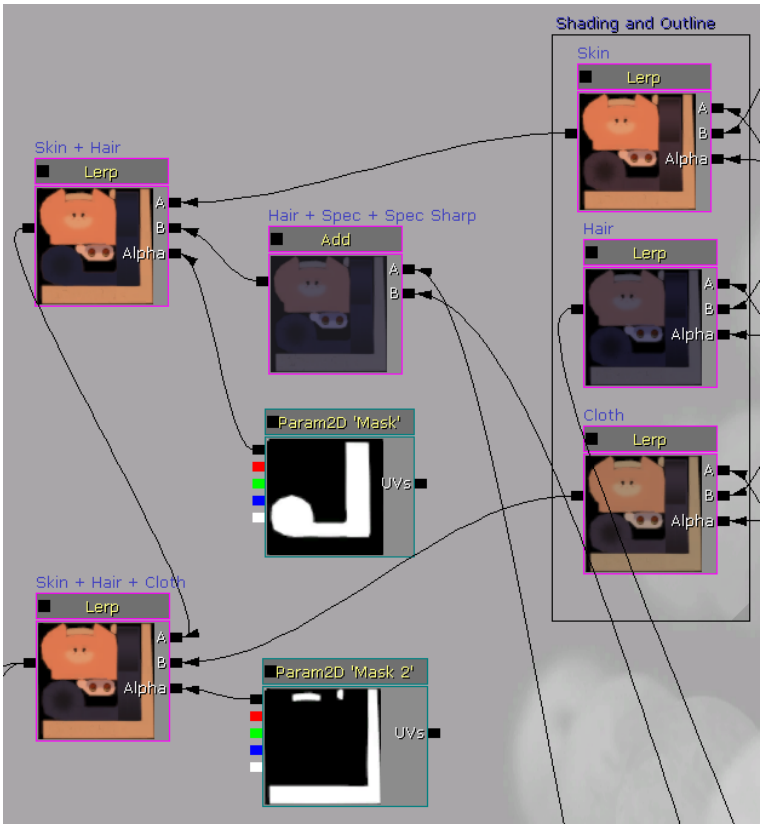


Figure 37: Individual masks in use.

This is done by drawing separate masks for each section of the model within Photoshop. This is made easier by having each section separate on the UV map, though on more complex models (such as a full figure) the mask will need to be more detailed. These two shapes (hair and clothing, the skin is assumed as the default) are then exported as masks to be used in the shader. This is done through the use of linear interpolate again, with the necessary mask being linked to the alpha node.

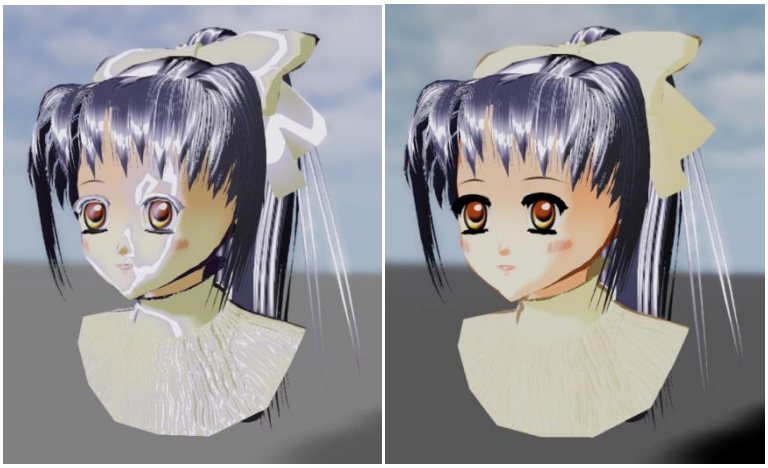


Figure 38: (Left) Outlines and highlights before masking.
(Right) Outlines and highlights after masking.

7.2.7 – Shadows.

Up to this point, within the preview window of the material editor, the shader appears to work fine. When tested in a true environment however, there is a noticeable issue concerning the shadows in the scene, as can be observed below *[fig 39]*.

Figure 39: Default shadowing.

Any shadows cast by a light source are rendered black, despite the application of the gradient ramp. The reason for this involves the specific way UDK works.

When applying diffuse based gradient ramps in 3D packages such as 3DS Max, the diffuse includes all lighting information, including both the lambert/phong/etc and any shadows that are cast by light sources. Many real-time engines on the other hand, including UDK, treat both the lambert/phong/etc and the shadows as two separate processes. Because of this, shadows are uniform by default, with one light casting the same colour of shadow across the entire scene within its radius.

The first instinct typically is to open the environment settings and boost the environment colour/intensity values (powered by lightmass) in order to remove the black shadows. While this does generally brighten the shadows in the scene, any crevices in the scene will still produce noticeably black shadows, mimicking the way shadows appear in the real world.

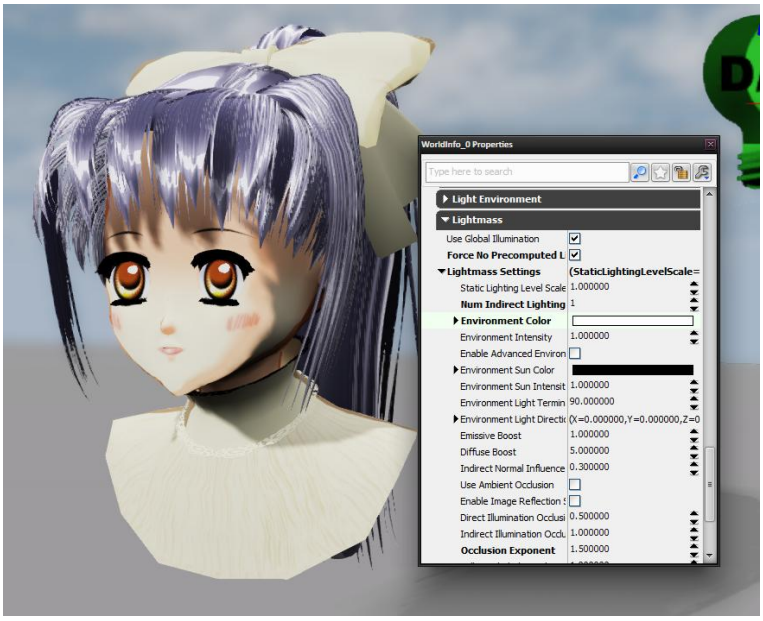


Figure 40: Boosting the environment values doesn't have the desired effect. Note how crevices (such as the neck area) still fade to black.

There is a workaround for this issue, however, a workaround which requires a good understanding of how both the lighting system and the shader system work within UDK.

First and foremost, UDK contains various types of light sources, such as spot lights, directional lights, and skylights. A spot light is a light source with a limited radius of effect, and is typically used for things such as lamps, candles, and other stationary objects which cast light. Directional lights have no radius of effect, instead casting an even amount of light across the entire scene. Directional lights are most commonly used for natural lighting such as sunlight. A skylight is a hemispherical light source which casts an even amount of light from both the upper and lower hemispheres separately. This light does not cast shadows, though it can add colour to existing shadows, and as such can be considered an ambient light. Originally this was used to add atmospherically appropriate ambient lighting to older Unreal Engine projects, but the addition of light mass has lead most to consider it redundant in this area. It is now included within UDK predominantly for backwards compatibility with older files. In spite of this however, many UDK users still find the skylight to be useful for a variety of situations, including improving the lighting on animated assets. In this case, the skylight can be used to boost the brightness/colour of shadows to the point where there is no longer any black, not even in the crevices of the scene, giving it a much more vibrant toon look.

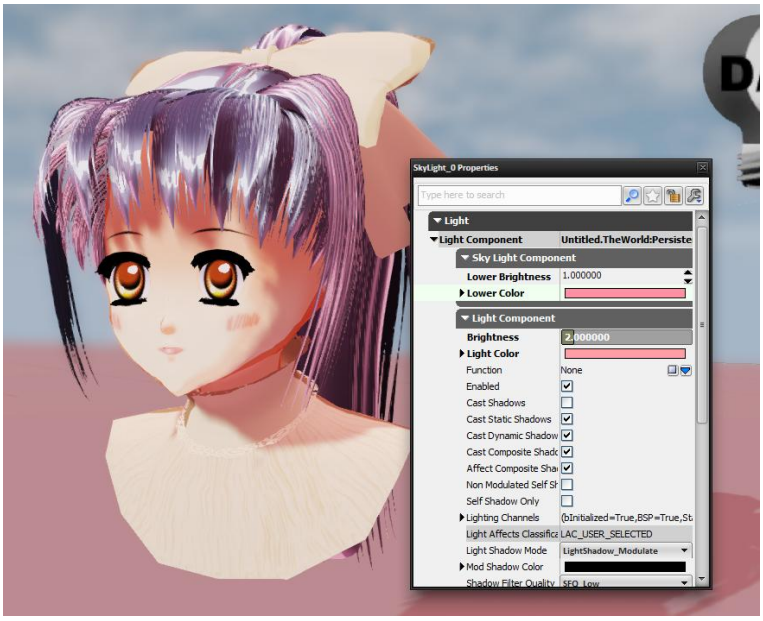


Figure 41: A skylight allows for the uniform boosting and colouring of the environment.

Although this solves the issue of the shadows being too dark, this doesn't solve the issue of colouring. While in some cases it is typically considered good practise for shading tones to be consistent (for example, red hues for warm scenes, and blue hues for cold scenes), and as such shadows of a single colour would not be an issue; for this particular style, it is necessary for the shadow colour to change depending on the surface. For example, the yellow clothing should not utilise the same pink shading the skin uses, and should instead use a darker shade of yellow to match the source material.

This problem can be solved with the custom lighting diffuse parameter, not to be confused with the custom lighting parameter. Although the custom lighting parameter affects how the objects looks in flat lighting, it is the custom lighting diffuse which affects how the mesh looks in shadow. Leaving this parameter blank would produce pure black shadows, whereas plugging the same node into both parameters (common practise when using the custom lighting mode) would provide dark shadows with consistent textures. It is also possible to add colour to these shadows by plugging a bright colour into this parameter, however, this would only provide subtle colour to the shadow, not boost the brightness of it.

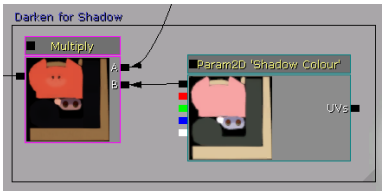


Figure 42: A new texture is used to apply multi-coloured shadows to the surface.

The shadow issue can be solved by combining both methods together. By using light shades of grey in the skylight, the brightness of the shadows can be boosted without adding colour. With the custom lighting diffuse, the colour of the shadows can be customised while providing minimal boost to the brightness. This allows for the customisation of any dynamic or baked shadows within the surface shader, giving the user the utmost control over the final look. Additional control over the colours of the shadows over different parts of the surface can be added with the use of a new "shadow" texture, which is then multiplied over the current shader setup for consistency. *[fig 42]*

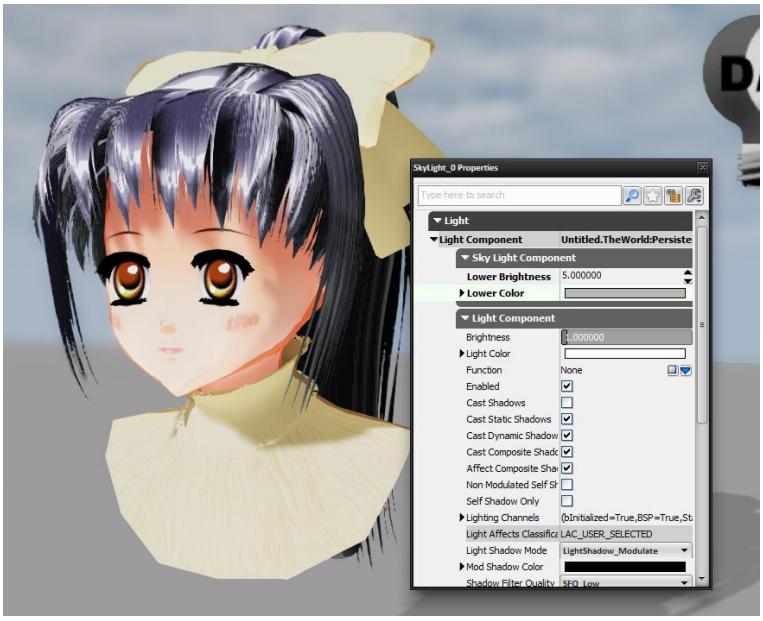


Figure 43: The final result.

7.2.8 – Normal mapping.

At this point, the shading looks fairly close to the source material, but in some areas (particularly the hair and clothing) the surface looks too flat. This is solved with the use of a normal map. In this particular case, two methods were used to generate the normal map.

The hair normal map was generated within Adobe Photoshop. A basic height map was created from a series of layered vectors and paint strokes, and then converted into a normal map with the NVidia normal map filter. These vectors were then re-used as part of the alpha map, ensuring consistency between the two.

The clothing normal map was generated within Mudbox, using digital sculpting methods to sculpt the necessary wrinkles into the clothing. The result was then baked into a normal map.

The two maps were then combined together into one normal map. Subsequent edits were made within Photoshop to make the effect stronger where necessary.

Figure 44: The normal map.

7.2.9 – Optimisation.

At this point, a close approximation of the style has been successfully achieved. Unfortunately though, the shader at this point has gone over UDK's imposed 12 texture limit, a limit put into place to ensure that artists don't create shaders that are too resource intensive. At this point it is important to look over the completed shader and find ways to optimise the texture usage.

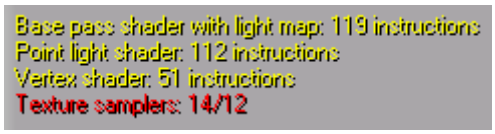


Figure 45: The texture samples error message.

7.2.9a – Optimising masks.

Firstly, the masks that are being used to separate the shading used on the skin, hair, and clothing should be optimised. Currently two separate textures are being used; one for the hair, and one for the clothing. Fortunately these can easily be combined into one texture through the use of the RGBA channels.

Whenever a texture is loaded into UDK's material editor, the user has the choice to use either the whole image, or any of the separate Red, Green, Blue, and Alpha channels. The most common use for this is to combine the diffuse/alpha into one texture, but it is also possible to store separate image information in each of the separate channels.

Figure 46: The "One-Texture Environment". (*Anonymous*, n.d.)

This was very important in the creation of the "One-Texture Environment" by Tor Frick, a sci-fi environment that uses only one 256 x 512 texture (*Anonymous*, n.d.). The texture included different details on each channel, including patterns, masks, numbers, logos, among other details. A complex shader structure was then used to take advantage of these channels, thus creating a highly detailed scene with only one texture. *[fig 46]*

To utilise this technique in the optimisation of this shader, the aforementioned two textures should be copied and pasted into the separate Red and Green channels of a new, blank texture (red for the hair, green for the clothing). When placed in UDK, the R and G channels of this single texture can now be used as separate masks, optimising two textures into only one.

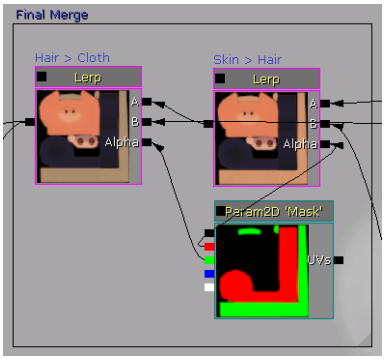


Figure 47: The optimised masks.

This is a fairly small optimisation, but an important one nonetheless, particularly with more complicated scenes and multiple characters.

7.2.9b – Optimising gradient ramps.

The most texture heavy section of the shader comes from the variety of gradient ramps used in order to achieve the toon shading effect, particularly in providing different coloured light based ramps to each part of the model.

When it is considered however that every light based ramp follows the exact same pattern (a smooth transition between two colours, all with the same strength), it becomes clear that these multiple textures are unnecessary, and that the effect can easily be produced with a single mask.

As each ramp only consists of two colours, each existing ramp can easily be recreated with the use of a linear interpolate. Linear interpolate has three inputs; colour 1, colour 2, and an alpha. By using a black and white gradient as the alpha, the user can then input the colours they need into the other two inputs as constant3s (RGB).

Here is a comparison of a coloured gradient ramp verses a linear interpolate ramp.

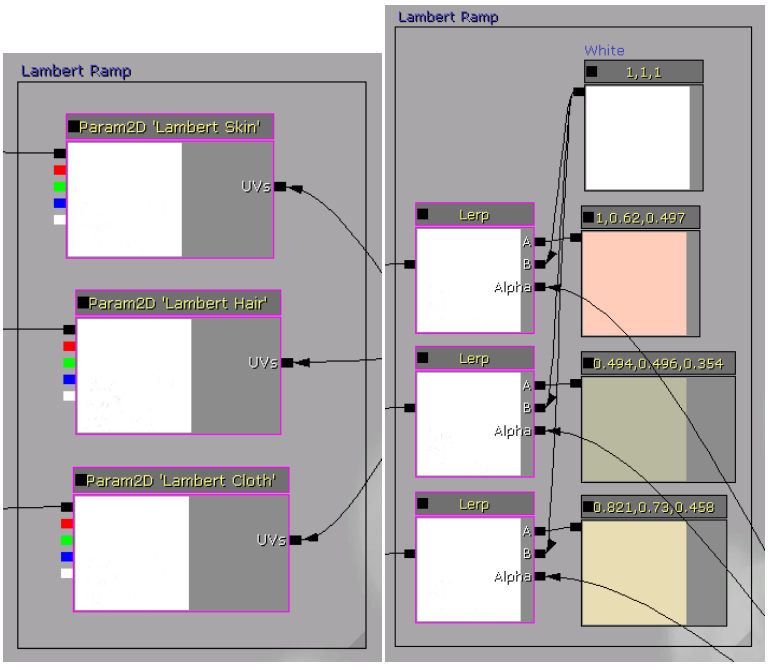


Figure 48: (Left) Un-optimised ramps requiring separate textures.
(Right) Optimised ramps utilising lerps, requiring only one texture.

Although it utilises more nodes, it does provide the user with two additional benefits. The first is that the colours can now be easily accessed and altered within the material editor without the use of an external program, and without needing to re-import the gradient ramp whenever a change is made to its colour. Another benefit is that this method can be easily and efficiently reused for every light based gradient ramp in the material, changing the input colours each time to match the different surfaces.

All of these subsequent linear interpolates share the same alpha, the black and white gradient ramp. As such, what previously consisted of six separate texture maps has now been successfully optimised into only one texture map.

The entire shader now only features three gradient ramps; the light based ramp, the sharp outline mask, and the sharp anisotropic mask. All of these are greyscale. With this observation in mind, it is clear that all three of these textures can now be combined into only one texture, utilising the aforementioned method of using the separate RGB channels of an image (section 7.8.1). These three textures have now been successfully optimised into only one texture map.

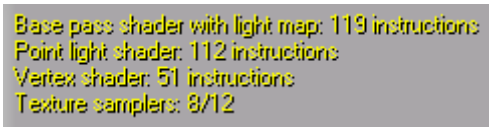


Figure 49: The texture samples are now below the limit.

The optimisation has been a success. While the shader was previously two textures over the Texture Samples limit, the shader is now four textures ***under*** the Texture Samples limit.

7.2.10 – Final result.

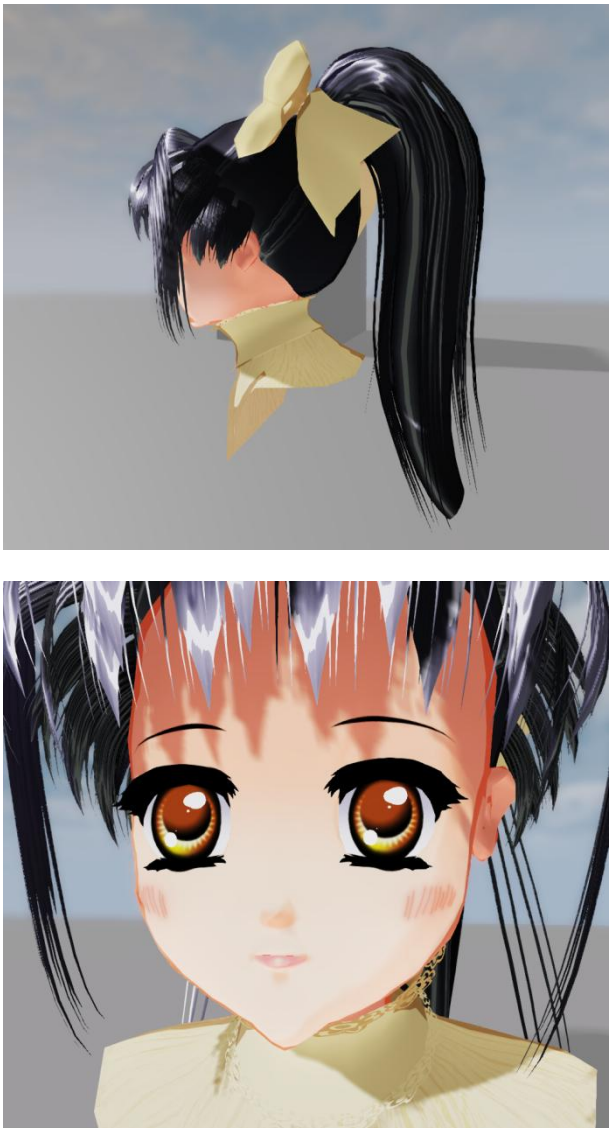


Figure 50: (Top to bottom) Four angles of the final result.

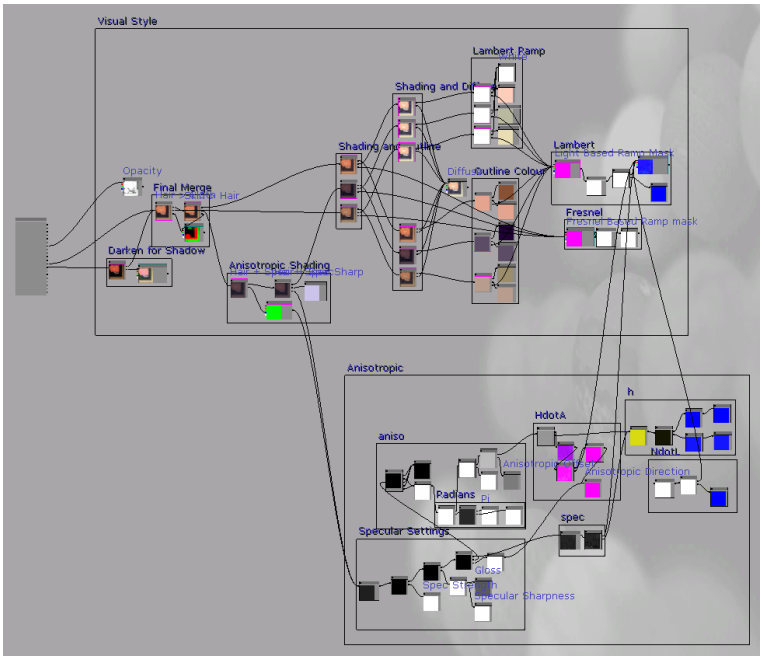


Figure 51: The complete shader network for the Visual Novel style shader.

7.3.0 – Experiment #2: Conclusion.

In section 6.2.0, it was hypothesised that the theory of the default bias was true, and that it would manifest itself in the Unreal Development Kit in some way. This has been proven true, the evidence of which is described in detail below. The other part of the hypothesis, however, was that the default bias would make itself known only during the later, more complex experiments. This has been proven wrong, however, as the default bias has instead made itself known multiple times as early as the first experiment.

In fact, it made itself known during the very first step of the shader's creation; the diffuse (section 7.2.1). As noted in section 7.2.1, UDK comes with a series of built in shader methods that allow users to quickly and easily create content suitable for photorealism, including phong, anisotropic, and subsurface scatter methods. Should the artist wish to break away from photorealism however, these will prove to be useless. Other 3D packages (such as 3DS Max) will give you open access to such pre-made rendering methods, allowing the artist to combine and alter these methods to create something new. In UDK, however, these methods are hardcoded into the engine, forcing the user to literally recreate these methods from scratch using the Custom Lighting parameter. This makes the creation of non-photorealistic shaders more complex than their photorealistic counterparts.

The default bias then provided a second obstacle through UDK's light mass. As noted in section 7.2.7, the UDK light mass system was created to mimic the way light and shadowing works in the real world, providing a direct benefit to photorealistic projects while proving to be a hindrance to this non-photorealistic project. This creates an odd situation where convoluted workarounds are required in order to make the scene look *more simplistic*.

To reiterate, the hypothesis was technically incorrect; although the default bias does exist, it appeared much earlier than anticipated. Furthermore, these particular examples of the default bias are ingrained in two of the most common steps in creating a shader; the shading, and the lighting. With this in mind it is safe to assume that these will become recurring obstacles in the subsequent

two experiments. Fortunately, the previously deduced workarounds can be utilised again in these scenarios.

As the default bias has now been proven to exist in UDK, all there is to do now is continue these experiments to see if the default bias manifests itself in any additional ways.

8.0.0 – Experiment #3: Manga style.

Manga is the word for "comic book" in Japanese, however outside Japan the word is commonly used to describe a style of comic book that either comes from an Asian territory (such as Japan) or is heavily inspired by traditional Japanese comics. Visually manga shares a lot of similarities with the anime style used in the previous experiment (section 7.0.0). While manga artwork may appear to be a very simple and easy to replicate style (especially when considering its lack of colour), there are various elements of the style that make it more difficult to accurately replicate than a typical toon image, especially in the case of real-time 3D.

In order to accurately and faithfully reproduce this style, a deeper understanding (and appreciation) of the art style is required.

8.1.0 – A brief history of manga.

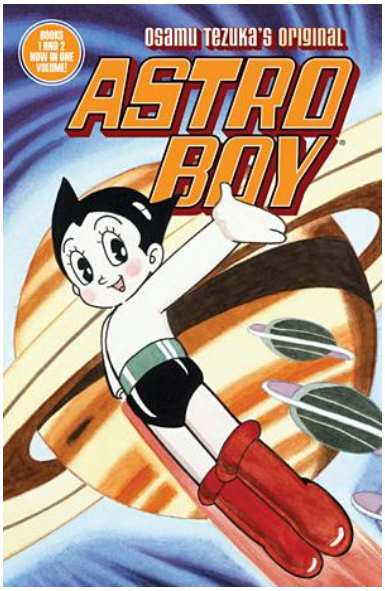


Figure 52: The front cover for Astro Boy. (Tezuka, 1952)

Although manga potentially has historic roots in scrolls telling stories using sequential images, the modern form of manga originated during the late 19th century, and is the culmination of various western influences.

> In fact, manga – that is, comics – might never have come into being without Japan's long cultural heritage being soundly disrupted by the influx of Western cartoons, caricatures, newspaper strips and comics. To deny this is to rewrite history. (Gravett, 2004, p.18)

These influences first took form in Japanese single-frame political cartoons, which were originally met with unease due to making critical comments about important people or events had been previously forbidden. It soon grew in popularity, with artists like Honda Kinkichiro taking this western influence and injecting it with Japanese pop-culture and references.

Soon, the concept of multi-panel comic strips started to overtake single panel comics in popularity in the form of newspaper strips. These were limited both in scope and content, as they tended to be read solely for brief humour. It wasn't until after World War II, during the U.S Occupation, that Japan became familiar with the concept of weekly and daily serials, particularly those produced by Disney. It was the large, expressive eyes and the simplistic art of Disney that inspired influential mangaka (Japanese comic artists) like Osamu Tezuka, the creator of characters such as *Astro Boy* *[fig 52]* and *Kimba the White Lion* (the latter of which being the inspiration behind Disney's *The Lion King*, bringing the inspiration full circle).

> One explanation for the popularity of comics in Japan… is that Japan had Osamu Tezuka, whereas other nations did not. Without Dr.Tezuka, the post-war explosion in comics in Japan would have been inconceivable. (Asahi, n.d cited in Gravett, 2004,p.24)

Tezuka helped shape the way modern manga is made in many ways. In terms of writing, he proved that comics could be emotional, and the Disney influence in his art helped shape the designs of many modern manga characters (most notably the use of large eyes). There is, however, one other common element to the manga style that grew predominantly out of convenience.

8.2.0 – Screentoning.

Just like the western comics that influenced them at the time, manga was originally printed only in pure black and white, due to colour printing being inaccessible. As such, many comic artists relied heavily on methods like cross-hatching or stippling in order to make their art more detailed, a process that was very time consuming.

It wasn't until the late 1930's however that both Japan and the west found new ways to add detail to their art, while simultaneously giving birth to two very different styles of printing.

Although the CMYK (Cyan, Magenta, Yellow, Black) method of printing had been around since the 19th century, it wasn't until the late 1930's that it became common within American comic books, starting the Golden Age of Comic Books. During this era, American comics utilised CMYK printing in order to provide detail to their comics that had never been seen before, providing much more visual clarity to their characters and worlds. This method of printing, however, also had some flaws, the main flaw being the price. Alex Woolfson, writer and publisher for OEL (Original English Language) manga studio Yaoi911, explains the expense of colour printing in more detail.

> There are good reasons why almost all manga is black and white. […] But the main reason is that it's tremendously more expensive to make a book in color. […] There's the expense of paying an artist to color your book, of course — not trivial. But particularly, it's the printing costs that add up fast. Paper for a black-and-white book runs through the press once to get that black ink — paper for a color book runs through 4 times to get each color for the CMYK inks. You get charged for that, and the extra ink, and the extra set-up costs — as you can imagine, it adds up fast. (Woolfson, 2006)

It is for this reason that the Japanese decided to forgo colour printing as their primary method, instead adopting a different technique of adding visual detail to their comics; screentoning.

Screentoning is a method of applying texture and shades to an image using only black ink. The process of screentoning involved the use of transfer sheets with the screentone patterns printed on them. These would be laid over the artwork in question, cut to size, and rubbed into the paper. (The Society for the Study of Manga Techniques (SSMT), 2003, pp.53-54)

Screentoning is an evolved form of cross-hatching and stippling; it allowed comic artists to add additional depth and detail to their art much faster, and due to the process still utilising black and white printing it was also much more cost effective (providing financial leeway for longer stories and more detailed art). As screentones also came in the form of pre-made patterns such as lace, petals, tartan, feathers, explosions, clouds, water, etc, comics that utilised screentoning were also able to

appear much more detailed than before, with artists no longer needing to painstakingly draw complex patterns. (SSMT, 2003, pp.34-35)

To summarise, while the west embraced this new method of colour printing in most published comics regardless of some of its flaws, Japan decided to take the methods it was already experienced with and find a way to improve it.

The two methods share one similarity, in that they have since become iconic styles. Many look back fondly at the western comics of the Golden Age, even going as far as to artificially recreate the flaws of the colour printing process at the time (such as the off centre sheets) in order to recreate the 'style' faithfully. Likewise, Japan's use of screentoning has also become an iconic part of the manga style, and is frequently the subject of mimicry amongst fans of the medium. Unlike the Golden Age style of colour printing however, screentoning is still used in most manga today, possibly due to a combination of both its iconic style, as well as it being cost-efficient.

8.2.1 – How screentoning works.

Screentoning is a very simple process that relies on an optical illusion. Screentone sheets consist of a very small pattern of shapes. When viewed from far away, the black and white pattern appears to form a shade of grey. This is similar to the process used by computer monitors and screens, using dots of RGB (red, green, and blue) that, when viewed far away, form different colours and subsequently images.

Different patterns will form different shades of grey, depending on how much white and black space remains in the pattern. To illustrate this point, the below examples *[fig 53]* show the dot pattern most frequently used in manga. On the left is a pattern consisting of a series of small dots spaced far apart. In this pattern, there is more white space than black, thus producing a light shade of grey. On the right is a pattern consisting of a series of larger dots placed closer together (to the point where they merge). In this pattern, there is less white space than black, thus producing a much darker shade of grey.

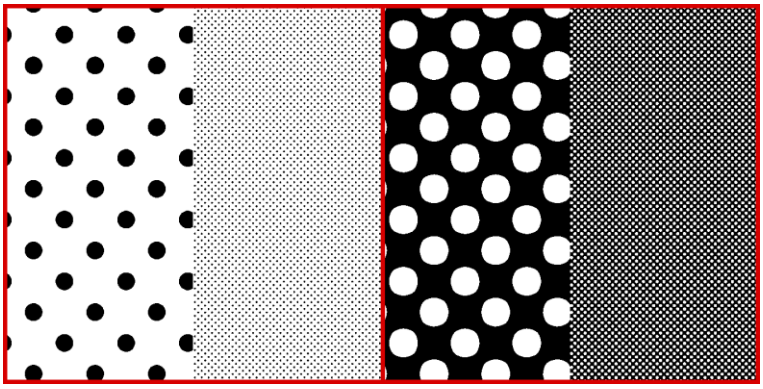


Figure 53: (Left) A light grey screentone, both close-up and from afar. (Right) A dark grey screentone, both close-up and from afar.

This process applies to patterns of any shape, such as lines, cross-hatching, etc. Although the dot pattern is the most common, most manga artists will use various alternate patterns, particularly to differentiate between surfaces such as cloth, hair, wood, metal, etc. This is because, as is visible from the above examples, the shade of grey does not appear flat; rather it has a visible texture to it. Although this is technically a flaw in the process, it has in itself become iconic with the style, with

artists taking advantage of it in order to add additional detail to their images (for example, a cross-hatching pattern could be good for woven fabric, whereas a more cloudy texture would work well for distressed leather/suede).

While these tones were traditionally applied manually with the use of wax paper, today these tones can easily be applied digitally in any art program. It is even possible to generate these patterns dynamically. With this in mind, it shouldn't be difficult to recreate this effect in a real-time environment. Similarly to the last experiment, where lighting information was used to generate a dynamic cel-shaded effect, the lighting information must be used this time to generate this screentone pattern.

8.3.0 – Dissecting the style.

8.3.1 – Shading / Diffuse.

In Moe (cute) manga, shading tends to be very simplistic. The faces have very little in the way of shading to them, with shading usually done purely in areas such as the nose and jawline. Shading only ever appears at extreme angles, and is usually done for dramatic effect. This tends to make most surfaces appear flat. The shading also tends to be very light, which helps to make the shading look softer.

Figure 54: Manga characters tend to have flat shading, but detailed shadows. (Yagami, 2002a)

The shadows, however, are much more detailed. If a character has a long fringe, the hair will cast detailed shadows onto the forehead. *[fig 54]* The same goes to other areas of the figure; the chin casting a shadow onto the neck, loose sleeves casting shadows onto the arms, etc. As for the colour, all shading is done in one tone; the shading does not get darker in areas that would realistically receive less illumination.

From this, it can be deduced that the lambert shading must be subtle at the most, if not completely removed. It may be necessary to remove all lambert shading and to rely only on dynamic shadows. Furthermore, it can also be deduced that an ambient occlusion map is unnecessary.

8.3.2 – Outlines.

The outlines in manga tend to be of an even width. Unlike the previous style which used very expressive outlines that varied in width, manga outlines tend to be much more simplistic. This is possibly to make shapes easier to distinguish from each other in the absence of colour.

Figure 55: Outlines are predominantly even, but may sometimes vary slightly in thickness. (Yagami, 2002b)

They may also have a noisy appearance, caused mostly by the texture of the paper. Depending on the artist, there may also be a few selected areas that have thicker lines than the rest. This can be either due to a stylistic choice, or due to human error.

The best method to recreate these outlines will be to combine two outline methods; fresnel based outlines, and post-process outlines.

The post-process outlines would generate even-width outlines across an entire scene via sobel edge detection, which is necessary to help clearly distinguish between objects. The fresnel outlines would be used to add extra emphasis to some of the curves in the surface, emulating the variations caused by human error.

8.3.3 – Hair Highlights / Anisotropic specular.

Not dissimilar to the source image used for the Visual Novel style shader, most manga characters have a sharp, horizontal white stripe across areas the hair. *[fig 56]* As noted before in section 6.1.4, this is directly based of how hair reacts to bright light in the real world, and is represented in rendering as anisotropic specular.

Figure 56: Manga hair highlights. (Inui, 2001)

In traditional manga, this effect was created in one of two ways; by scratching away any tone applied to the hair, or by applying white ink onto the shaded area (typically pure black hair). This tends to result in the shine being sharp and thin. In this case, the previously built anisotropic specular module can be reused and reapplied on top of the new shader.

8.3.4 – Textures.

There is one noteworthy practice that appears in many examples of manga, and this is the use of flat gradients. This effect is most commonly used on hair. In such cases, darker haired characters may have a vertical gradient applied to the hair as opposed to just a flat tone. *[fig 57]* There are three possible reasons for this. It could either be a part of its style, an efficient way to apply some simplistic shading, or a way of emulating a person's roots being darker or lighter than the tips of their hair (such as if they dyed or bleached their hair).

Figure 57: The gradient helps provide simplistic shading for dark hair. (Yagami, 2002c)

To emulate this, a gradient effect needs to be applied in such a way that it doesn't wrap around the object in 3d space, instead appearing completely flat to the screen. Furthermore, the gradient effect needs to be restricted to only the hair section, and not affect any other part of the model.

8.4.0 – Replicating the style in UDK.

8.4.1 – Screentone effect.

As the shader relies heavily on this effect in order to work, this effect should be made first, with all other parts of the shader being catered to work with this effect afterwards. It is necessary to devise a shader network that can take a pre-determined pattern (such as a dot pattern) and dynamically alter it based on the value (brightness) of the surface.

8.4.1a – Faking the effect.

It is possible for this effect to be faked through the use of multiple tiling textures, each having a pre-created pattern for each required shade. These textures could be applied straight onto the surface using the shadows as a mask. Although this may appear to be a simple solution, there are numerous flaws with this method that make it impractical for larger projects.

The first and most obvious issue is the use of textures. Although this method would work with no issues on untextured surfaces, applying a texture onto a surface using this method would result in a style dissonance; the shadows would be screentoned, but textured details would not. This becomes particularly problematic in areas such as the eyes, face, and clothes of a character, as well as any other surface that requires textures for additional detail (such as foliage, ground, sky, particle effects, etc). Getting textures to work with pre-made screentones would require additional texture masks for light, medium, and dark sections of the texture.

Then there is the issue of the patterns themselves. Assuming the user is only using one pattern, a dot pattern, they would require at least five versions of the dot pattern (two light, one medium, and two dark). Using additional patterns, a practise which is common in manga, will require additional textures.

It is apparent that attempting to fake the screentone effect would not only be impractical, but would also be impossible to produce in UDK due to the large amount of textures needed. As observed in section 7.8.0, UDK has a limit on how many textures can be used in one shader. By dynamically generating the effect as opposed to faking it, the amount of required textures is cut down by a large

amount. This would also allow for changes made to a surface's texture or colour to be quick and efficient, as the effect will dynamically adjust to such changes.

8.4.1b – Creating the dynamic screentone.

In order to develop a dynamic screentone effect, the shader will need to be able to compare the base pattern with the colour of the surface, altering the look of the pattern depending on the result. The optimum way to do this is through the use of a gradient map.

Figure 58 (left) shows a tiling gradient map that goes from white to black. The white areas represent the brightest point of the surface, and the black areas represent the darkest. This texture essentially contains every dot pattern possible for each shade of grey.

How it works, is that the shader will compare this gradient map to the surface colour at a particular point. For example, say the surface is a medium grey. The shader will compare that shade to the gradient map, and find all of the pixels that match that particular shade of grey.

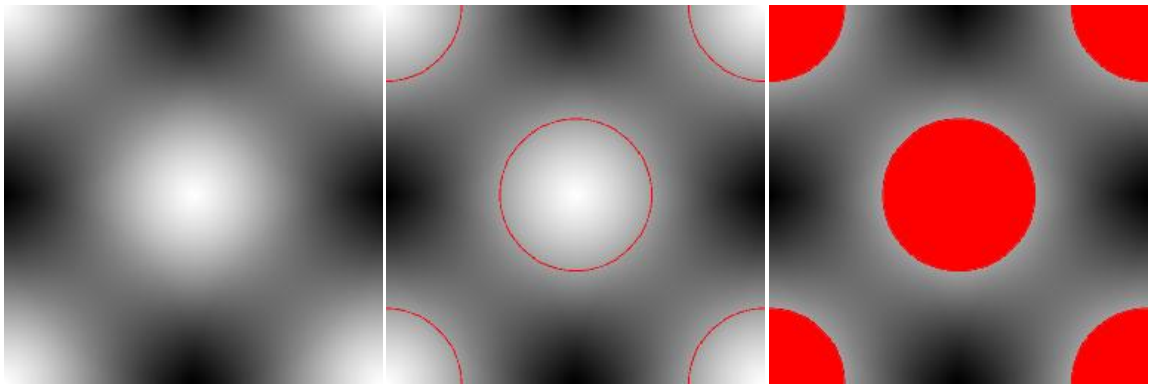


Figure 58: (Left to Right) Gradient map, initial comparison result, final comparison result.

As shown in figure 58 (middle), this comparison results in a circle of pixels that match the same shade of grey. The shader can subsequently fill this circle in by including anything "greater than" the current shade (keeping in mind white is 1, and black is 0).

As shown in figure 58 (right), this forms a complete dot. The shader can then colour this dot black, and apply it as a pattern. As the shader continues along the surface, additional comparisons are made based on the frequency of the pattern. If the surface is a darker shade of grey, the comparison will find areas of the gradient map that match that particular shade, forming a larger dot. Lighter shades will produce smaller dots. Subsequently, white will produce no dots at all, and black will cause the entire area to be filled in black.

The final result, when applied as a flat pattern across the surface, will be that of a dynamic screentone.

8.4.1c – Flat pattern.

Unlike a traditional texture, a screentone should not conform to the shape of the surface. It should instead be completely flat, as if a sheet of screentone paper has been applied to the screen. To do this, the screen position node is used. This node outputs the screen-space position of every pixel on the screen. By default this outputs three values; height, width, and depth. As depth isn't needed for this effect, a component mask is used to isolate the R and G values, outputting only the width and

height. Plugging this into a texture through the UV slot would produce a texture that appears flat to the screen, not conforming to the mesh.

In order to make it tile, the width and height should be multiplied by a numerical value. Preferably the user should be able to set the amount of tiling for the width and height separately for more control. To do this, two constants are used; one for horizontal tiling, and one for vertical tiling. These constants are then appended together to form a constant2.

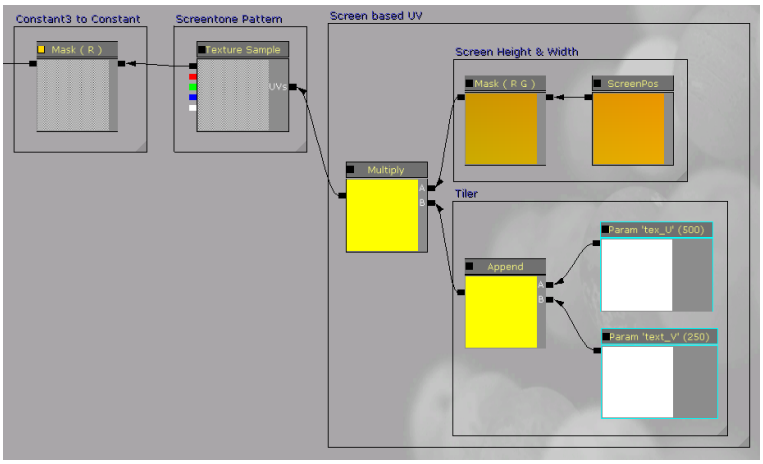


Figure 59: The screentone texture using the flat tiling method.

This is done due to the way maths is applied in UDK. For example, if a constant3 (***2,2,2***) is multiplied by a single constant (***2***), each separate value in the constant3 is multiplied by that single constant (resulting in ***4,4,4***). If that same constant3 (***2,2,2***) were multiplied by another constant3 (***1,2,3***) however, each value is treated as a separate equation (resulting in ***2,4,6***).

The same applies to constant2s. When the R and G values were previously isolated from the screen position node, what was originally a constant3 (R,G,B) was converted into a constant2 (R,G). Multiplying this by another constant2, such as the previously appended constants, would result in each value being treated as a separate equation. This allows the user to easily alter the tiling amount of the screentone pattern when necessary.

This turned out to be necessary, as the texture becomes stretched horizontally on a widescreen aspect ratio, requiring the user to set the horizontal (x) tiling to double the vertical (y) tiling.

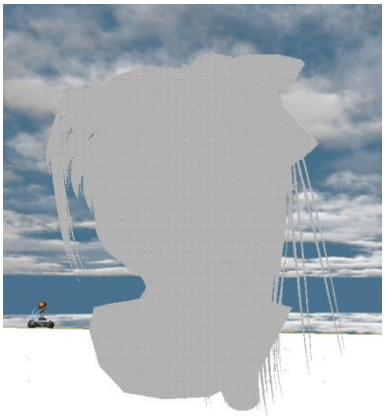


Figure 60: The screentone texture applied onto a mesh.

8.4.1d – Applying the screentone.

As established in section 9.1.2, the screentone shader needs to be able to compare the colour of the surface to the pixels of the gradient map, outputting any result which is equal to and greater than the current shade of grey. The best choice for this would be the if node, which compares two inputs and provides different outputs depending on the result. As noted in the last experiment (section 7.4.1b), the if note has three possible solutions; if input A is equal to input B ($A = B$), if input A is greater than input B ($A > B$), and if input A is smaller than input B ($A < B$).

There is an issue however; the if node only allows constants to be compared, whereas the diffuse texture and the screentone pattern are constant3s. Being textures, they contain RBG colour information, even though they only use a greyscale palette. When all three RGB values are the same, it produces a shade of grey.

This problem is easy to fix however, by using a component mask. A component mask can take any constant with more than one value, and mask specific values. For example, running an RGB texture (a constant3) through a component mask will add the capability to individually mask one or more of the RGB values. This component mask was previously used to isolate the R and G values of the screen position node in section 9.1.3, converting a constant3 (RGB) into a constant2 (RG).

In this case, the component mask can be used to convert a constant3 into a single constant. As the RGB values of the greyscale texture are the same, the individual R, G, and B components will also look the exact same when separated. As such, the component mask can be used to isolate one value (R) from the greyscale texture, converting it into a single constant that can be used with the if node, without affecting the appearance of the texture. This is done for both the diffuse texture and the screentone pattern. Both of these values can now be utilised in the if node.

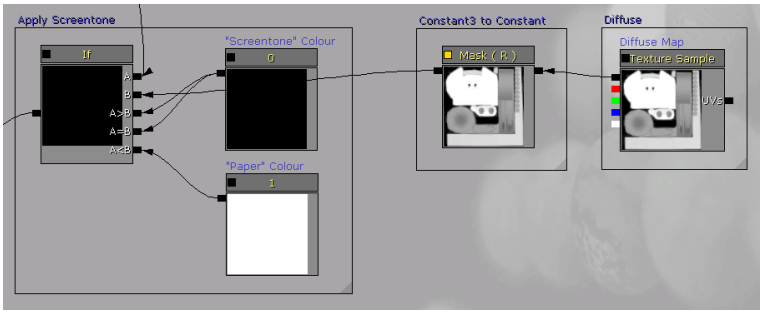


Figure 61: Applying the screentone effect to the diffuse texture.

With the diffuse as input B, and the screentone pattern as input A, two additional constants are needed; one that is pure black (0), and one that is pure white (1). The former is used to provide the colour of the screentone pattern, while the latter is used to provide the colour of the base (or the 'paper'). The black constant is then plugged into both the "*A=B*" input (this draws the initial circle) and the "*A>B*" input (this fills in the circle). The white constant is then plugged into the "*A<B*" input (the paper colour). This produces a fully working screentone shader.

Figure 62: The initial screentone effect.

8.4.1e – Texture compression.

Although the screentone shader works, there are two issues with the effect. The first and most obvious is the noticeable tiling effect of the screentone (also known as a moiré effect), the second being the high contrast of the effect. Fortunately, both can be solved through the use of the texture compression settings for both the diffuse texture and the screentone pattern texture through the Texture Properties.

Texture compression settings are very important, as the correct compression settings can help reduce the rendering cost of each texture without sacrificing quality. These settings can also ensure that each texture behaves correctly in certain situations.

Firstly, the screentone texture's settings will be changed in order to fix the tiling issue. Within the Texture Properties, there are options specific for tiling textures such as Preserve Border (R, G, B, or A). Using one of these will automatically improve the appearance of the tiling effect on most surfaces, but not all.

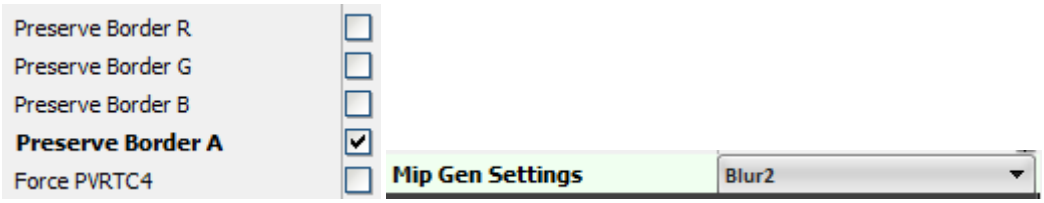


Figure 63: (Left) The Preserve Border options. (Right) The Mip Gen Settings.

Right at the bottom of the list are the Mip Gen Settings. Mipmaps are pre-generated, optimised versions of a texture, used in order to determine how a texture looks at smaller sizes. These are very important in video games, as textured objects are often viewed from both a short distance and from afar. UDK however has the ability to automatically generate these mipmaps through a variety of different settings, such as blur and sharpen. In this case, Blur2 provides the best result; it blurs out the edges of the tiled screentone pattern, reducing the appearance of the tiling. Higher settings end up blurring the texture too much, obliterating most details. Blurring the screentone patterns also has the unintended side effect of reducing the contrast of the effect, making it much less harsh.

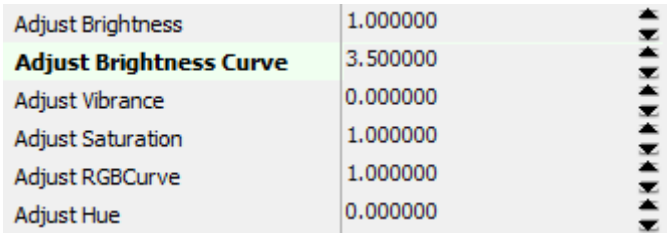


Figure 64: The texture properties used on the diffuse texture.

It is at this point that another issue makes itself known; the diffuse texture is too light, with most of the details not being picked up by the shader. This could be fixed by altering the brightness/contrast of the texture within an image editor, however this requires the user to re-import the texture multiple times until the right look is achieved, which can take a while depending on the size of the texture and the speed of the machine. Fortunately though, UDK has the ability to make such changes directly within the engine, by editing the Texture Properties. Within these properties the user can adjust both the Brightness and the Contrast (the latter being labelled Brightness Curve) quickly and efficiently, with nearly instant results. In this case, simply setting the Brightness Curve to 3.5 (originally 1) provides enough contrast for all of the texture's details to be picked up.

Figure 65: (Left to Right) Effects of the Preserve Border, Blur 2 Mip Gen, and Brightness Curve texture properties, culminating in the final screentone effect.

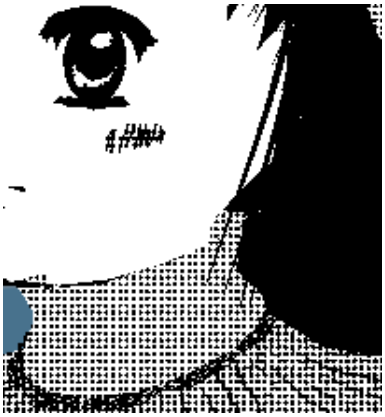


Figure 66: A closeup of the final screentone effect.

Now that the screentone effect has been produced, it is now possible to focus on the other aspects of the shader.

8.4.2 – Diffuse.

For the purpose of this shader, it has been decided that the lambert effect isn't necessary for the shader. Instead, the shader shall rely solely on UDK's shadowing system. Dynamic shadows are sharp by default, providing the cel-shaded style needed for this effect. It should also be noted that unlike the visual novel style shader (section 7.0.0), which required multi-coloured soft shading, this effect is much more simplistic in terms of colour usage. As such, lambert shading is unnecessary.

8.4.3 – Shadows.

It is at this point that the same default bias obstacle from section 7.2.7 is encountered again, producing realistic shadows that are unsuitable for the style. In order to achieve the right kind of shadows, the same techniques from the section 7.2.7 should be used. The user should first start with the combination of a Dominant Directional Light (specifically the Moveable version, which has the ability to cast sharp, dynamic shadows, necessary for detailed shadows on characters) and a skylight. Just like in section 7.2.7, the skylight is there to help even out the appearance of shadows, removing the natural black gradient UDK's lightmass applies in order to mimic real world lighting.

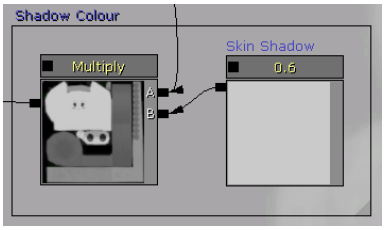


Figure 67: Shadow colour setup.

The next step is to apply the screentone effect to the shadows. Similarly to section 7.2.7, this is achieved with the use of the custom lighting diffuse input, which directly affects the tint of surface shadows. This is done by taking the initial diffuse texture and multiplying it by a single constant, which gives the surface a darker appearance. This is then fed through the screentone process again (component mask -> if), producing the effect of screentoned shadows. Unfortunately, in doing this, two problems are encountered.

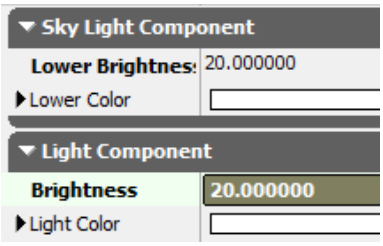


Figure 68: Skylight settings.

The first, and most important, is the look of the shadows in the final scene. Although the shadows are more even, they still retain some of their initial colour. *[fig 70, left]* This produces a medium shade of grey in areas where it should be white, breaking the pure black and white style of the screentone effect. This problem is easily fixed however by increasing the brightness of the skylight's upper and lower hemispheres to their full brightness. This has the effect of bleaching out the grey areas of the shadow (the "paper"), while leaving the black areas of the shadow (the screentone) intact.

The reason for this is due to the shadow effect being multiplied over the scene. As the brightness of the skylight is increased, the amount of times the shadow is multiplied over the surface colour is also increased. This will eventually lead to the surface becoming a stark white. Black, however, has the numerical value of 0. Anything multiplied by 0 will still remain 0. As such, anything that is pure black (such as the screentone effect) is left untouched by the skylight. This wouldn't work with the previous Visual Novel style shader, as the entire scene requires colour that would otherwise be bleached out. Without the need to worry about colour, this obstacle becomes much easier to overcome.

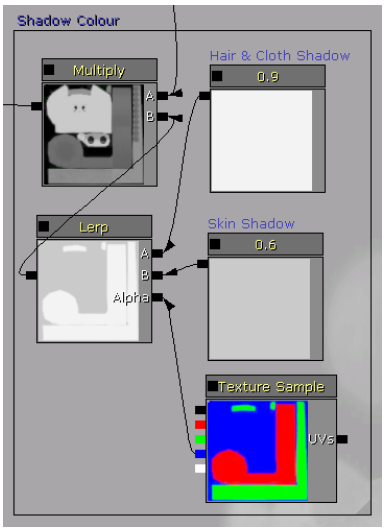


Figure 69: Applying multiple shadow colours using a mask.

The second problem is much simpler to fix. Whereas the chosen shadow colour works well with the skin, it appears too dark on the clothing. To solve this, a texture mask is used to apply a much lighter shadow colour to these sections, providing the user with more control over the final appearance. *[fig 69]*

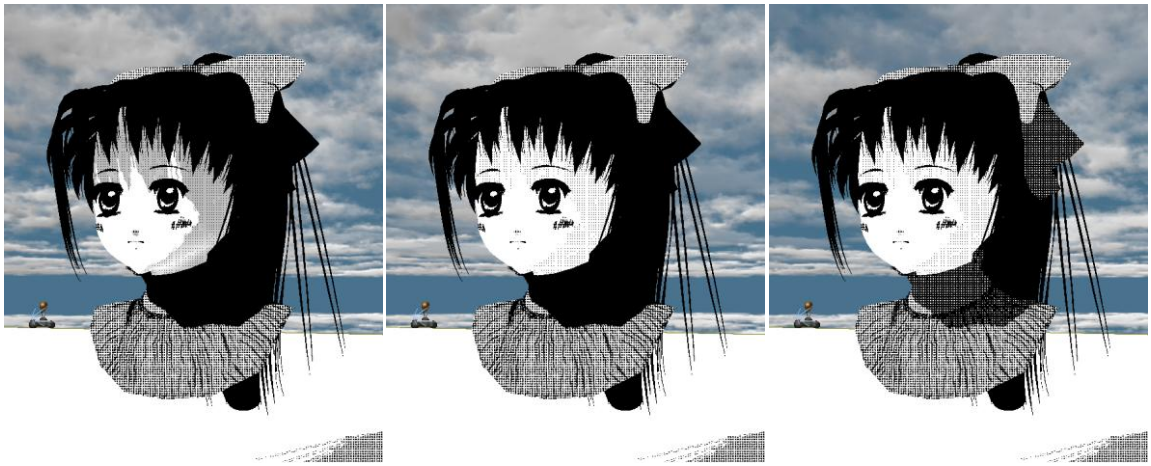


Figure 70: (Left to Right) Initial shadows, shadow consistency fixed, shadow colour fixed.

8.4.4 – Specular.

Yet another recurring default bias obstacle from the previous experiment, this time the anisotropic specular from section 7.2.4. The anisotropic specular effect must be recreated from the ground up to compensate for UDK's own anisotropic specular effect being hard-coded into the engine. Fortunately, the previous anisotropic specular network created in section 7.2.4 can be re-used with only minor alterations. By altering the gloss and sharpness parameters, the shader will produce the thin highlight that is needed without the use of a gradient ramp.

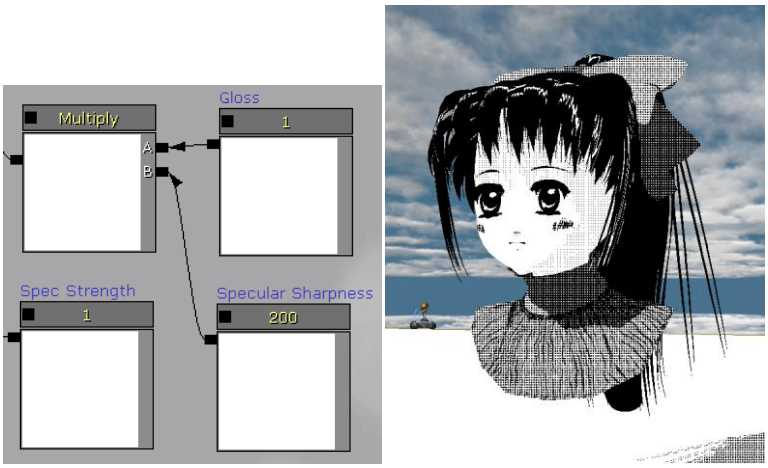


Figure 71: (Left) Anisotropic specular settings.
(Right) Final anisotropic specular effect.

8.4.5 – Normal map.

The same normal map from the previous experiment is being used to provide extra detail to the character's shading. It should be noted though that for the most part this normal map's effects are subtle due to the lack of lambert shading (thus increasing the need for the painted on wrinkles in the clothing). Despite this the normal map is still important, as it is imperative to the anisotropic

specular effect; it deforms the highlight in such a way that it appears jagged, similar to most manga characters of this type.

8.4.6 – Gradient effect.

As discussed previously in section 8.4.4, some manga artists apply flat gradients to their artwork, usually as a way of applying some soft lighting quickly and efficiently (most commonly on detailed surfaces like hair). For the purpose of this experiment, this effect will be recreated for use on the model's hair.

First, a texture with a vertical gradient is needed, going from white (top) to black (bottom). Next, the gradient should be applied as a flat texture, as it is part of the screentone effect. This is done using the same methods used in the production of the screentone effect (section 9.1.3), masking the R and G values of the screen position node. The gradient effect shouldn't tile like the screentone, however, and as such should be supplied its own separate horizontal and vertical tiling parameters, tweaked in order to ensure that the gradient covers the entire screen.

After that, this effect is applied to the surface by using the add node, adding the gradient effect over the texture (as the original hair texture is already quite dark, the gradient will be used to lighten the top of the hair). This applies the gradient over the entire surface, which is fixed through the use of a linear interpolate with a texture mask, isolating the effect to the hair. All of this is added in directly before the screentone effect.

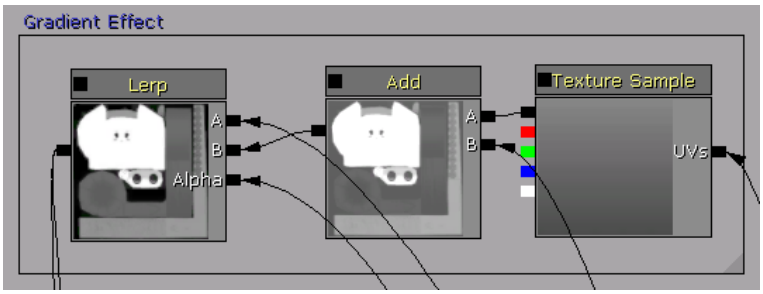


Figure 72: Initial gradient shader network.

At this point, the gradient is applied and works as it should. The effect is too strong however; the top is too bright, and it goes dark too quickly. *[fig 74, left]* To fix this, another linear interpolate is added before the add node, using the vertical gradient in the alpha input. In inputs *A* and *B* are two constant3s (RGB, required when using linear interpolate); one for the top of the gradient, and one for the bottom of the gradient, both set to different shades of grey. This reduces the strength of the effect.

There are two reasons why the linear interpolate is the best method to use in this situation.

1. The first is that it allows the user to quickly and easily alter the gradient effect within UDK, as opposed to having to edit the gradient texture in an image editor and then subsequently re-import it.

2. The second is versatility. Should the user want to reuse this gradient effect on a different surface, the effect will need to be made stronger or weaker depending on the surface. Rather than using multiple different gradient textures, which would take up more space and

memory, the shader can instead use just the one texture, with the user editing the constant3s when needed.

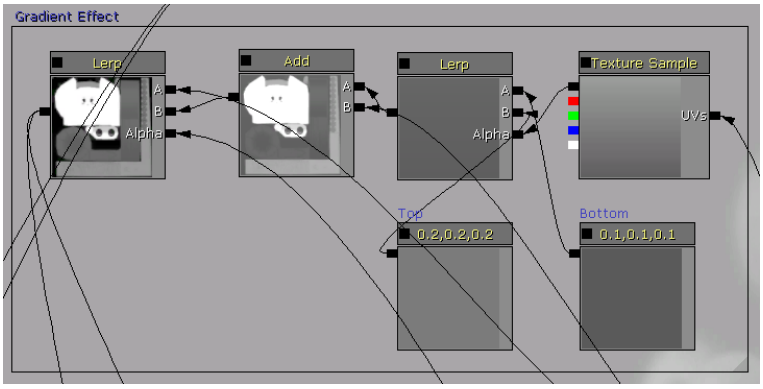


Figure 73: The final gradient shader network, utilising lerp.

The final result is that of a flat gradient effect that is isolated to the hair. *[fig 74, right]* Since the gradient is applied before the screentone effect, the gradient is also accurately screentoned. Not only is this good for simulating the manga style, but the smooth gradient is also good for showing the power of the dynamic screentone effect.

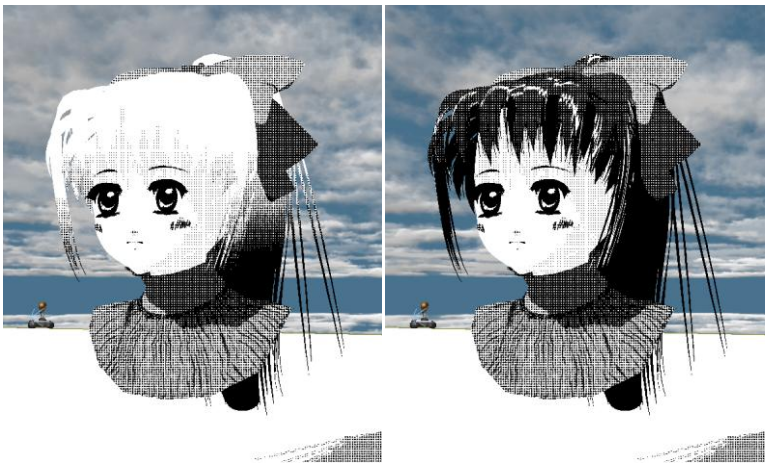


Figure 74: (Left to Right) The initial gradient effect, and the final gradient effect.

8.4.7 – Outlines.

As previously observed in section 8.4.2, the outlines in manga are for the most part uniform, with very little variance in width. They are also of a uniform colour; black. It is for this reason that the shader will utilise both fresnel based outlines and post-process outlines. The fresnel based outlines will provide the slight variance of width in areas such as the nose and jaw line, while the post-process outlines will provide the uniform width outlines needed for the majority of the figure (as well as the surrounding scene).

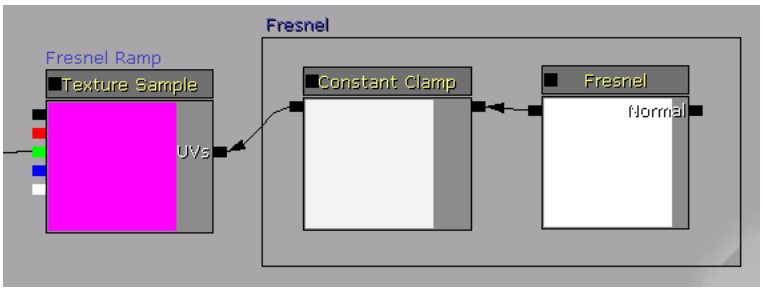


Figure 75: Fresnel shader network.

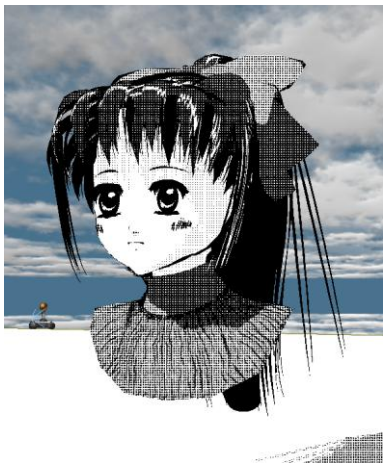


Figure 76: Fresnel outline result.

First, a fresnel node is plugged into the UV input of a gradient ramp, which outputs a sharp outline. This is then combined with the rest of the shader to add some subtle, ink outlines towards the edge of the shape. The post-process outlines are then subsequently used in the scene itself.

The post-process outlines are generated using a process called sobel edge detection. Sobel edge detection typically works by going through an image pixel by pixel, generating an image based on any abrupt changes to hue, brightness, or saturation. When applied on a photograph (say, as a filter in an image editing tool such as Photoshop), this will not only draw outlines around separate objects within an image, but also around shadows, highlights, etc, as it relies solely on the colour information. Within UDK however, a scene depth pass can be used to generate a much more accurate outline that only appears around the edges of a mesh, and not around any shadows or textured details.

The shader itself is impossible to reproduce, as the official documentation provided by Epic omits important information on how particular nodes are used. Fortunately, Epic have a pre-made sobel edge detection shader available for download as part of their **Unreal Development Kit Gems**, a collection of tutorials and pre-made resources intended to help beginner UDK users produce their own projects with ease. (*Anonymous*, 2011) Unfortunately though, the sobel edge detection shader provided by Epic is far from perfect, suffering from some graphical glitches.

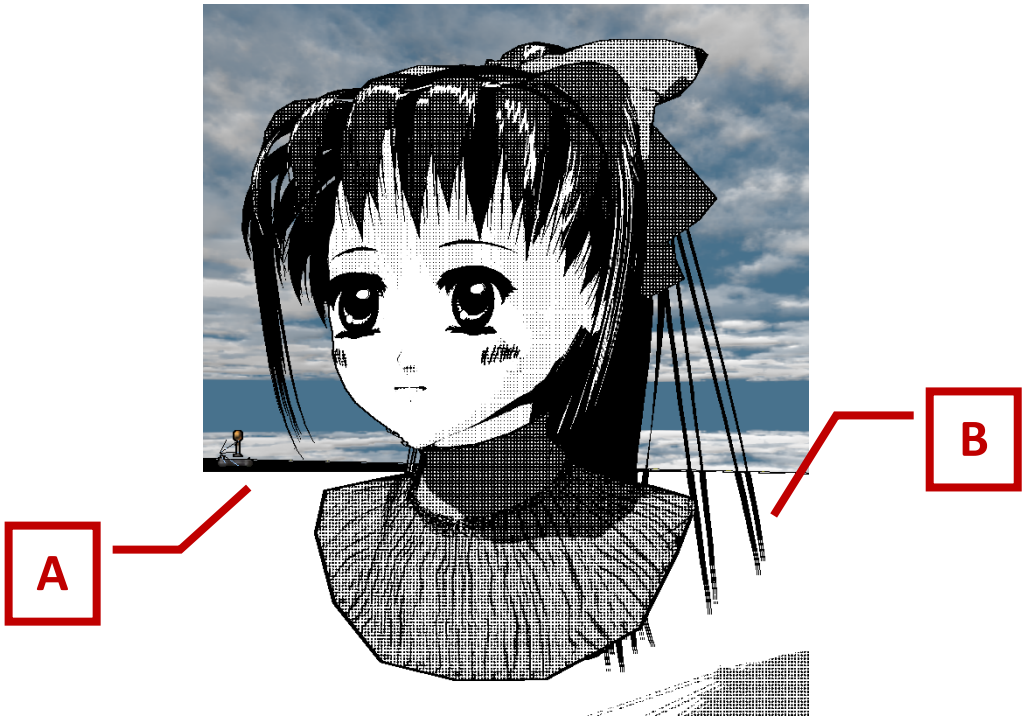


Figure 77: The initial sobel edge detection result.

The first issue can be observed in the background *[fig 77, A]*. Surfaces further away from the camera (especially flat surfaces) become entirely painted in black. Secondly, whereas mesh outlines appear fine in front of the skybox, in front of other meshes in the scene the outlines appear twice as thick as if they have been drawn twice. This is especially noticeable on the thin strands of hair coming out of the character's ponytail *[fig 77, B]*.

Unfortunately these flaws aren't completely unavoidable, though the appearance of them can be reduced. The first issue happens due to the maximum distance being too low. As the shader gets closer and closer towards the maximum distance, it starts becoming less and less accurate, to the point where details start to blend into each other, causing some areas (particularly flat areas) to become shaded in blotches of black. This appearance can be reduced by altering the max and min distances to be spread further apart. By setting the maximum distance to a larger value, details in the distance start to become more detailed. This isn't enough however, as the minimum distance also needs to be altered into a smaller value (preferably a negative value), or else details closer to the camera may become lost.

The second issue is also caused by z-depth, and takes place in two parts. When the sobel edge detection detects a large difference in two pixels, it produces the outline. While the shader is using the z-depth of the entire scene to generate the outlines, it is only using the z-depth as a standalone image; the sobel edge detection does not take the actual depth of an object into consideration. As such, it produces pixels on *both sides* of the difference, inside and outside, as it doesn't differentiate between either sides of an edge. This is similar to the issue that causes most objects to have a dark halo around them when using Screen Space Ambient Occlusion; as these are real-time shaders, they are by nature significantly less accurate than their pre-rendered counterparts.

Scalar Parameter Values	... (Overriding 5/10)
AbientDarkness	0.200000
AmbientBrightness	1.000000
Contrast	1.500000
DepthExclusionDistance	32768.000000
MaxDistance	1000000.000000
MinDistance	-679385.125000
MinimumIntensity	0.050000
ResolutionX	1920.000000
ResolutionY	1080.000000
SobelPixels	1.125000

Figure 78: Sobel edge detection settings.

The second part of the problem lies in the depth exclusion. As it isn't possible to exclude specific objects/surfaces from a post-process shader within UDK, the skybox/skydome is also affected by the sobel edge detection, which can produce unsightly results. To stop this from happening, the shader uses depth exclusion to erase outlines that are a certain distance away from the camera (presumably only the skybox) after they have been drawn. This not only stops drawing outlines across the skybox, but also includes any of the aforementioned outlines drawn on the outer side of any detected edges, essentially removing half of the outline. This makes outlines appear thinner when in front of the skybox, and thicker when in front of other objects. Unfortunately again, there is no way to completely remove this problem, although its appearance can be made less severe by increasing the *resolution X* and *resolution Y* values (in this case, to 1920 and 1080 respectively, also known as 1080p resolution), effectively making the outlines thinner. The issue is still noticeable, though not as much as before. *[fig 79]*

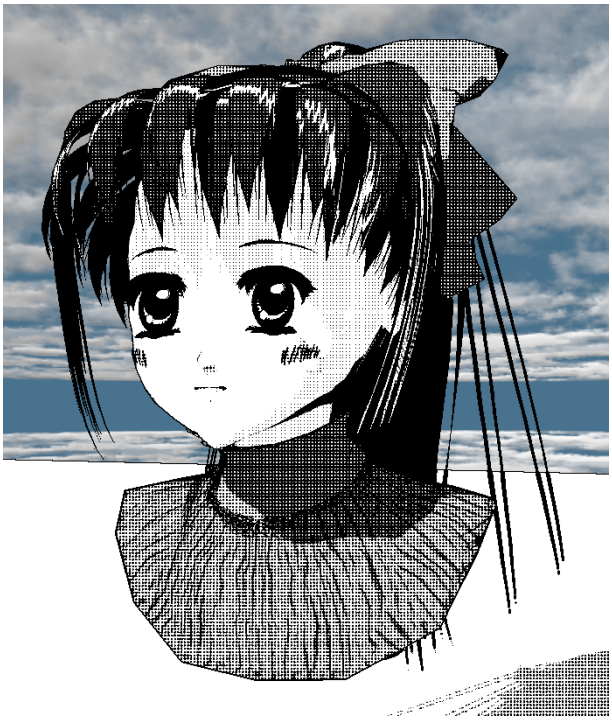


Figure 79: Final sobel edge detection effect.

8.4.8 – The sky.

To complete the look of the entire scene, the screentone effect has been ported into the stock skybox shader. This was done by placing the previously established screentone process after the entire skybox shader process. While this does apply the dynamic screentone effect to the standard dynamic clouds, there are two issues; the sky looks too dark to be used in a daytime scene and the light blue horizon has been left intact. The latter is due to the horizon being a part of the environment settings, and as such is separate to the shader. This horizon can be disabled, but doing so subsequently reveals the lower half of the skybox, revealing the fact that the sky effect is vertically mirrored *[fig 82, top]*. Because of this, it is necessary to somehow recreate the horizon effect in shader form.

This was done using Object Orientation. This expression outputs the upwards direction of the object in world space. Combined with a Vector Transform, this will provide a visual representation of the *x*, *y*, or *z axis* in world space, starting from the centre of the object and ending at the edge of the object. When used along the *z axis* (used as the vertical axis within UDK), this creates a gradient that looks not dissimilar to the horizon effect. A constant clamp is used to make this gradient appear slightly harsher. *[fig 80]*

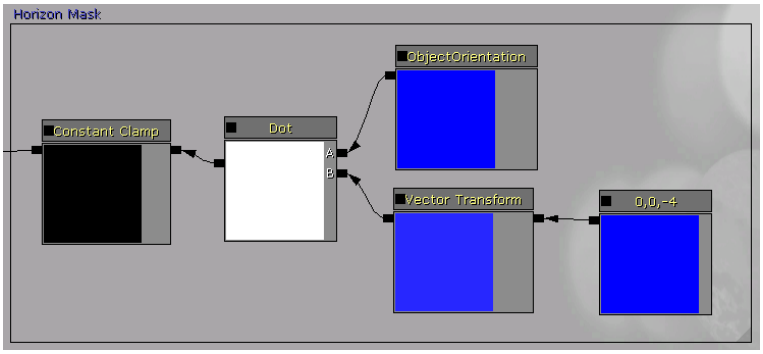


Figure 80: Generating the horizon.

The finished Object Orientation gradient is then used as the alpha in a linear interpolate. Both of the other inputs are constant3s; input A being the Horizon Brightness (pure white), and input B being the Sky Brightness (dark grey). *[fig 81]* The result is then added over the existing sky shader, producing a soft horizon while also dynamically brightening the rest of the sky.

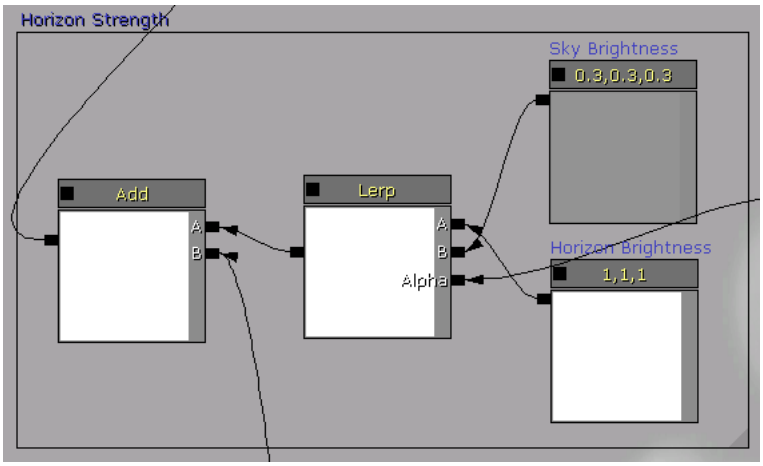


Figure 81: Applying the horizon to the rest of the sky shader.

The final result is an animated skybox that utilises the screentone effect without sacrificing the horizon effect.

Figure 82: (Top) The original sky shader.
(Bottom) The screentoned sky shader (with horizon).

8.4.9 – Final result.

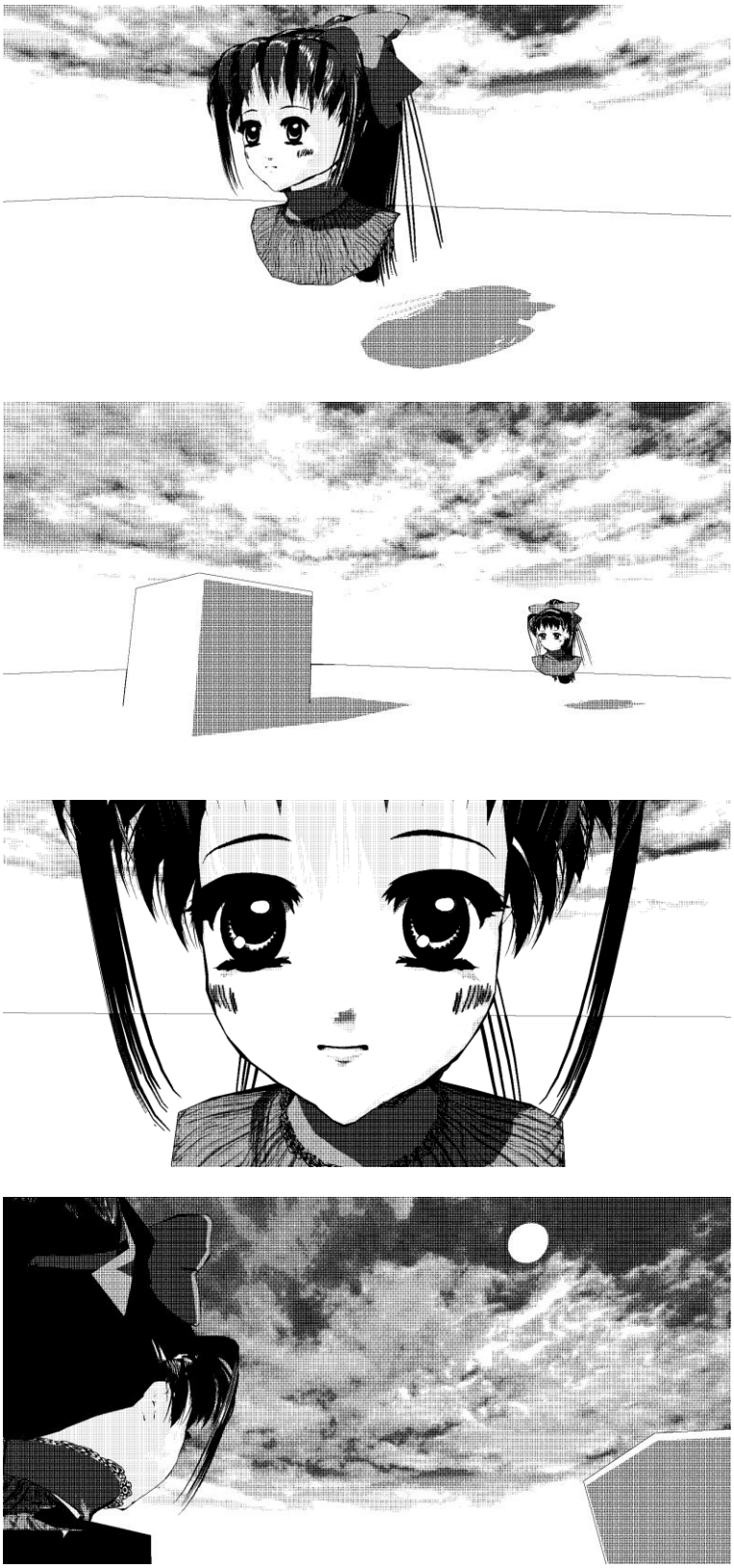


Figure 83: (Top to Bottom) Four angles of the final result.

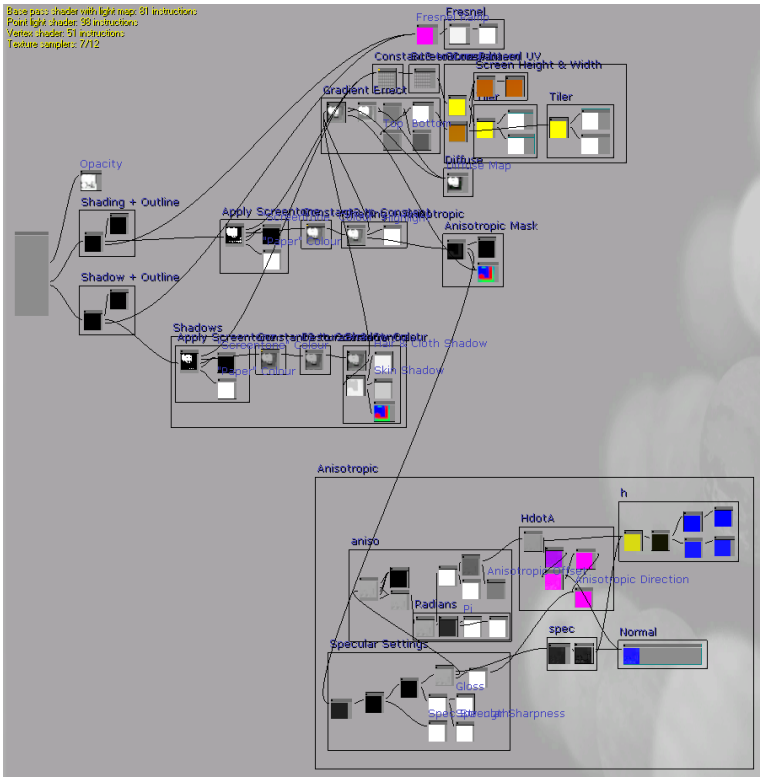


Figure 84: The complete shader network for the manga style shader.

8.5.0 – Experiment #3: Conclusion.

As expected, two of the already discovered obstacles caused by the default bias have made themselves known again; the lightmass shadows (section 8.4.3), and the anisotropic specular (section 8.4.4). Fortunately, as also expected, the previously established work-arounds in sections 7.2.7 and 7.2.4 could be re-used with only minor alterations in order to suit the new art style.

While there was an additional obstacle in the form of the sobel edge detection post-process effect (section 8.4.7), this was not an obstacle caused by the default bias, but rather due to the limitations of the shader itself (similarly to the SSAO shader discussed in section 7.2.3).

Aside from those examples there have been no additional obstacles encountered, including those caused by the default bias.

9.0.0 – Experiment #4: *Fable 2* concept art style.

This final experiment, while not in itself an iconic style such as manga, is much more relevant to the issue at hand; realising the conceptual artist's stylistic vision into 3D. It is also a much more practical experiment in the sense that it is based on a pre-existing game, specifically Fable 2 (mentioned in section 5.2.1). Specifically this experiment will use the concept art of the character Barnum as a reference. *[fig 85]*

Figure 85: Official concept artwork for the Fable 2 character Barnum. (Lionhead Studios, n.d.)

9.1.0 – Dissecting the style.

9.1.1 – Shading & Tones / Diffuse.

The colour scheme for this style can be described as 'earthy'; the colours used for both the base colours and the shading are warm yet desaturated, with the only hint of vibrancy coming from the secondary blue light source. Unlike the anime style, which is consistently bright even in shaded areas, this style uses quite dark shading in areas that would realistically be dark. Similar to most 'toon' shaders however, black is never used.

As for the style of shading, from far away the shading does appear somewhat smooth. There is no noticeable sharpening as with most cartoon styles, instead opting for a much more even transition from light to dark. Upon closer inspection however, these transitions were painted by hand in a fashion that mimics traditional painting, with a noticeably cloudy appearance.

9.1.2 – Tone variation and details / Ambient occlusion.

As this character is much more detailed and closer to a realistic human than an anime character, it is as expected that the shading is detailed enough accommodate what appears to be ambient occlusion (shadowed areas caused by crevices). It should be noted though that not only are these areas represented by the use of darker colours, but they are also represented by the use of simplistic, scribbled shading.

This is a by-product of how the concept art was produced. The original design was sketched out onto paper in a very fluid style, getting all of the necessary details down without any of the shading (as this would be added in later). Some details, such as the sunken in eyes and cheeks, are represented by some faint sketched shadows to help emphasise the depth of these areas. This sketch was then taken into a digital painting program to be painted over, during which all of the colours and detailed shading is added in.

This sketching appears predominantly in sunken in areas of the face such as the eyes, cheeks, and ears. These are areas that would be affected by a very strong ambient occlusion effect. There is also some faint sketching across the rest of the figure, though these sketches are more to do with the texture of the surface as opposed to the shading of the surface (for example, numerous sketches appear on the clothes and apron to emphasise how messy they are).

9.1.3 – Highlights / Specular.

The highlights in this image are nowhere near as bright or as vibrant as with the previous examples; they are much more subtle, giving the impression of many of the materials (such as his goggles) being made of a dull metal. His skin in particular uses specular highlights similarly to how they appear in the real world; they are dim and small, giving the impression of greasy skin. This could be accomplished with a simple phong specular.

9.1.4 – Outlines.

The outlines in this image are subtle. Rather than the thick, prominent outlines that typically appear in regular toon images, the outlines are much softer, with the image relying mostly on the detailed shading in order to differentiate between areas. The outlines really only appear in a few areas where some details could get lost in the lighting/shadowing, such as the mouth, nose, and ears. Any outlines that do appear are dark, following the same warm colour scheme as the rest of the image, and have a soft, free-handed look to them. This can be accomplished with a soft fresnel effect.

9.1.5 – Paper texture / Overlay.

The entire image appears to have a generic grunge pattern laid over it, giving it an aged and weathered appearance. Such methods are regularly used to easily apply an additional sense of detail over an otherwise finished image and can help to further emphasise the mood of the image, helping to give the image more personality. As this is a generic flat texture laid over the image, the effect can either be applied as a flat surface effect (similar to the gradient effect used in the manga style shader in section 8.4.6), or as a flat post-process effect.

9.2.0 – Replicating the style in UDK.

For the purpose of this experiment a stock 3D model has been used, specifically one generated using the MakeHuman tool by the MakeHuman Team. A generic male model was generated within MakeHuman, which was then further altered using the sculpting tools within Zbrush (also used for

texturing and normal mapping) to provide a much closer resemblance to the character of Barnum before being exported into UDK. All models generated using MakeHuman are covered by the Creative Commons Zero (CC0) license, and as such can be freely redistributed and used in any project regardless of whether it is commercial or non-commercial. (MakeHuman Team, 2012)

9.2.1 – Diffuse.

The diffuse begins with the standard lambert shading, produced using a dot product of both the light vector and the normal vector (in this case, the normal map). This creates smooth, lambert based shading. Furthermore, the shading in the original source image has a cloudy texture. To reproduce this, a suitably cloudy texture is used.

Similar to the screentone style from before, this tiling cloudy texture is rendered flat against the screen using the same method described in section 8.4.1c. This texture is then added to the lambert shading, giving the lambert shading a suitably cloudy appearance.

This cloudy texture, however, appears too uniform. *[fig 86, left]* While some of the brush strokes appear flat in the source image (particularly in large areas), in some areas the brush strokes are clearly following the contours of the shape.

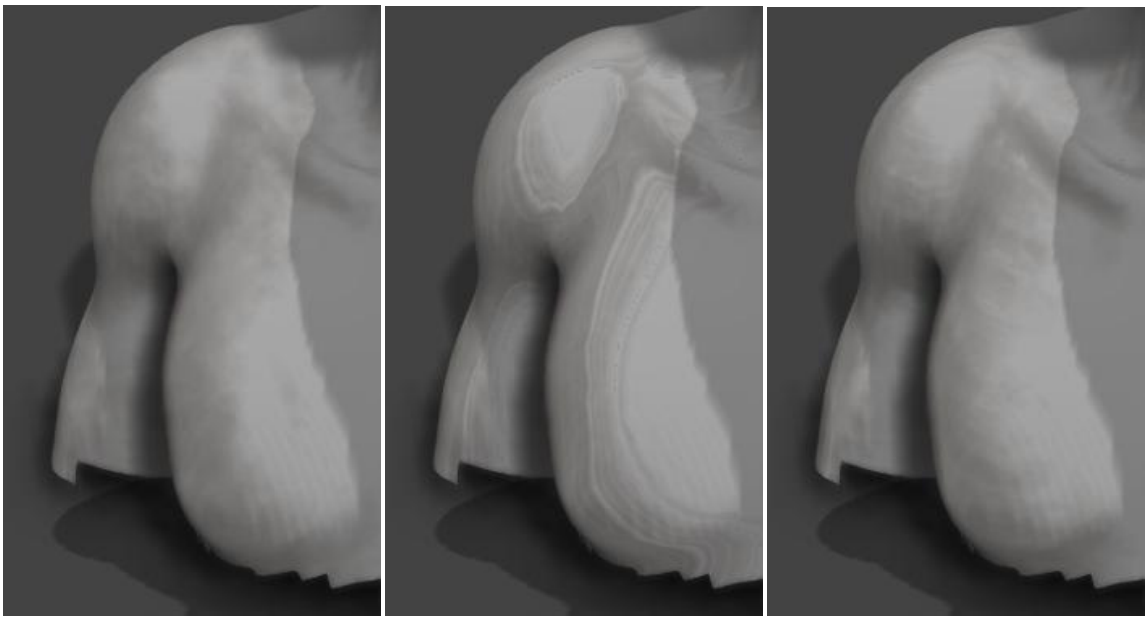


Figure 86: (Left) Screen position tiling is too flat. (Middle) Fresnel tiling is too streaky. (Right) A combination of the two.

This effect can be reproduced to an extent with the use of fresnel. Applying a fresnel to the UV input of the cloudy texture in place of the screen position method will cause the texture to distort at the edges of the figure, giving the edges a streaky appearance not too dissimilar to paint strokes. This looks too streaky however, and also has the side effect of distorting the texture across flatter surfaces. *[fig 86, middle]*

The issue can be fixed however, by combining both methods. This is done using a linear interpolate. As noted before in section 7.8.2, a linear interpolate can blend between two inputs based on an alpha. If a constant is used as an alpha, there will be a consistent blend between these two inputs. For example, if *input A* were red, *input B* blue, and the alpha a constant set to 0.5, the output would be purple.

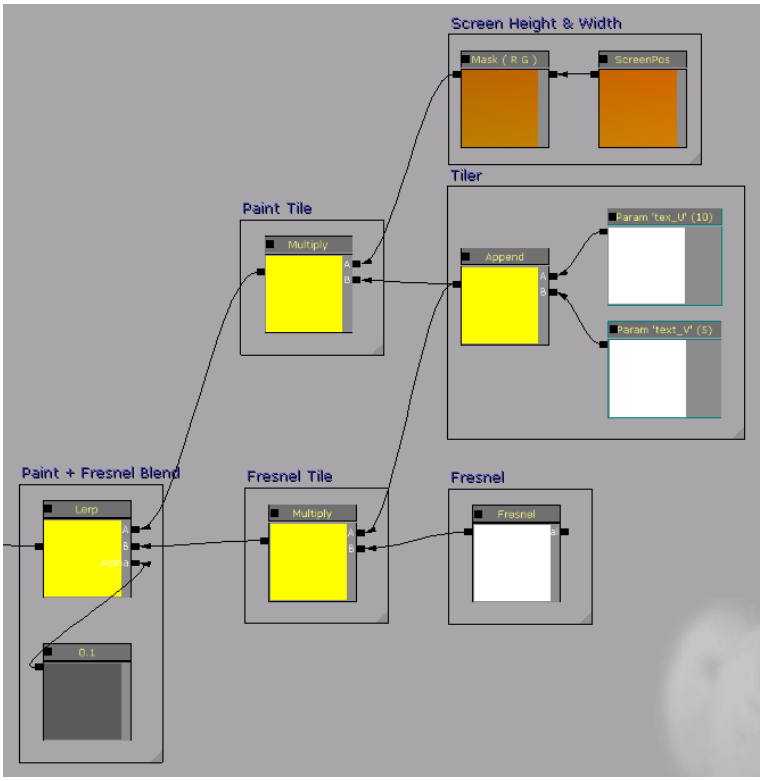


Figure 87: The complete tiling method, combining screen position with fresnel.

In this case, the screen position method is set as *input A*, and the fresnel method as *input B*. A constant is used to blend between these two methods until there is an adequate amount of edge distortion without sacrificing the flat application. *[fig 86, right]* In this case, the constant has been set to 0.1.

Now that the effect has been made, it can now go through the usual process of using a gradient ramp to add colour. UDK will not allow this however. Only a float2 output, such as a fresnel (two variables, normal vector and camera vector) or a lambert (also two variables, normal vector and light vector), can be used in the UV input of a texture. It should be noted that a typical UV map also has only two variables, height and width (a 2D representation of the mesh). By adding the lambert shading (a float2 node) to the cloudy texture (a float3 node with three variables, Red, Green, and Blue), the process is converted from a float2 output into a float3 output, which cannot be used as a UV input. To fix this, a component mask is used to output only two variables (in this case Red and Blue), converting it back into a useable float2.

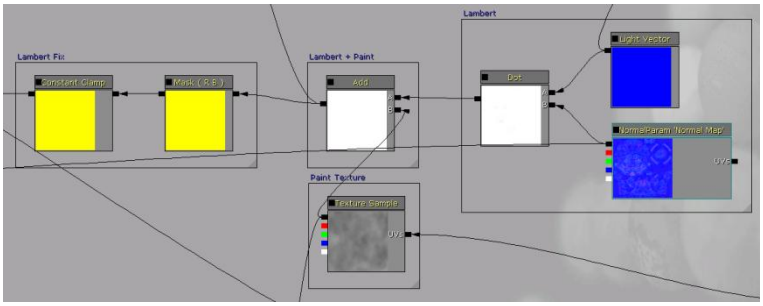


Figure 88: Combining the paintbrush effect with the lambert, and then using the component mask.

With that fixed, the shading process is connected to the gradient ramp, producing the necessary colours for the painterly effect. This is then multiplied over the diffuse map similar to the Visual Novel style shader (section 7.1.0).

Figure 89: (Left) The final diffuse in basic lighting conditions.
(Right) The final diffuse in 2-point lighting mimicking the source image.

9.2.2 – Outlines.

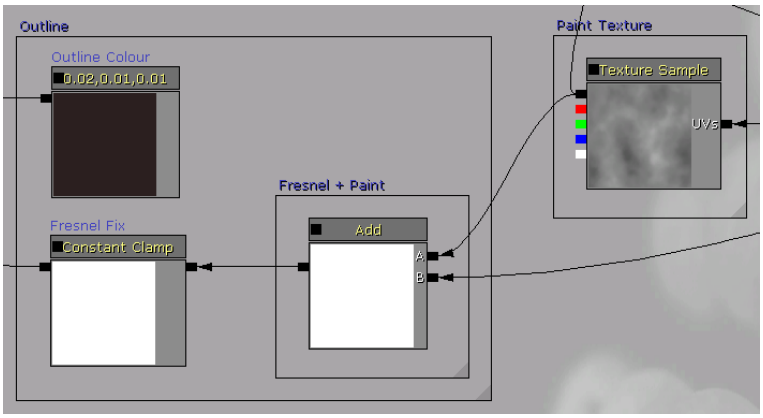


Figure 90: Creating the outline.

Fresnel will be used to produce the outlines. Unlike in the previous experiments however, the fresnel does not need to be sharpened; the original source material uses a soft, subtle outline, so the standard fresnel can be used on its own. To reduce memory consumption, the same fresnel node used in the creation of the cloudy effect can be used. The cloudy texture is then added to the fresnel, so that it matches the style of the rest of the surface.

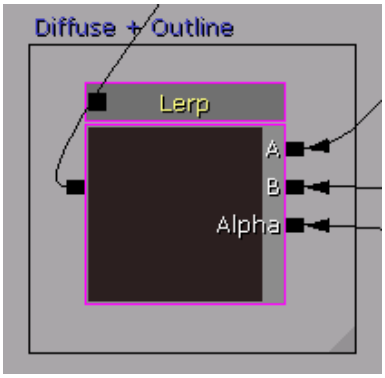


Figure 91: Combining the outline with the diffuse.

Finally, the outline is combined with the diffuse via a linear interpolate, using the fresnel as the alpha and the needed outline colour is input B (in this case, a constant3). As can be observed below *[fig 92, right]*, the produced fresnel effect has the unintended (but welcome) side-effect of applying a light paintbrush texture to the entire surface, giving it a rougher appearance not dissimilar to the source image.

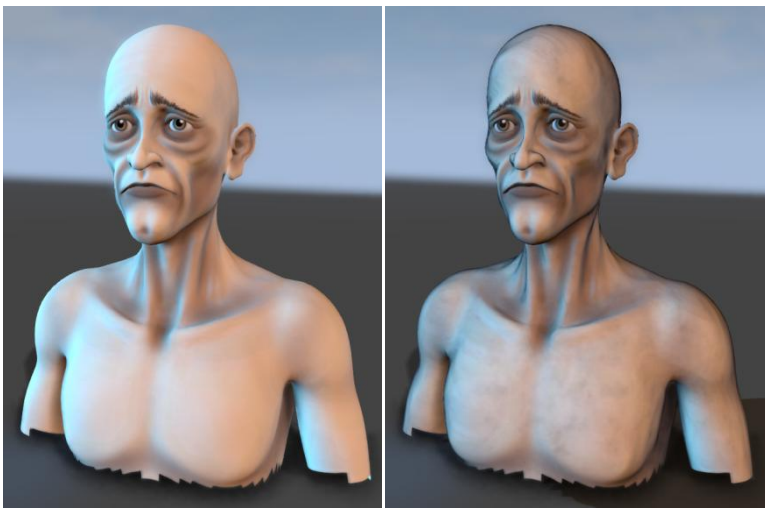


Figure 92: (Left) The surface without outlines. (Right) The surface with outlines.

9.2.3 – Specular.

As noted before in section 9.1.3, a phong specular will be used to apply a glossy sheen to the skin. UDK does have phong shading built in but, just like the anisotropic specular, this is only in the form of a lighting mode. When using the custom lighting mode, the phong specular must be recreated. Fortunately, Epic details how this is done in their official documentation, using it as an example of how custom lighting effects differ from their built in counterparts. (Wright, n.d.)

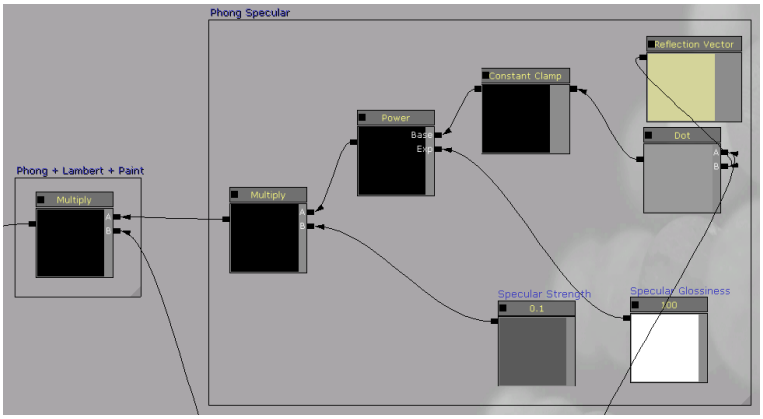


Figure 93: The phong specular shader network.

The phong is recreated firstly by producing a dot product of both the light vector and the reflection vector. This produces a very basic phong highlight. After being clamped to avoid glitches, this is then fed through a power node. This node has two inputs, *Base* and *Exp*. The power node takes whatever is plugged into the *Base* node and multiplies it by itself *Exp* times. In this case, this has the effect of making the phong specular sharper and smaller, giving it a much glossier appearance.

Finally this output is multiplied by another constant, which has the effect of making it either brighter or darker, providing control over the strength of the specular. This output is then multiplied once more by the lambert shading to ensure that highlights appear only in lit areas. This completes the phong shader network. *[fig 93]* Once this network is completed, it is then added to the combined diffuse and fresnel. *[fig 94]*

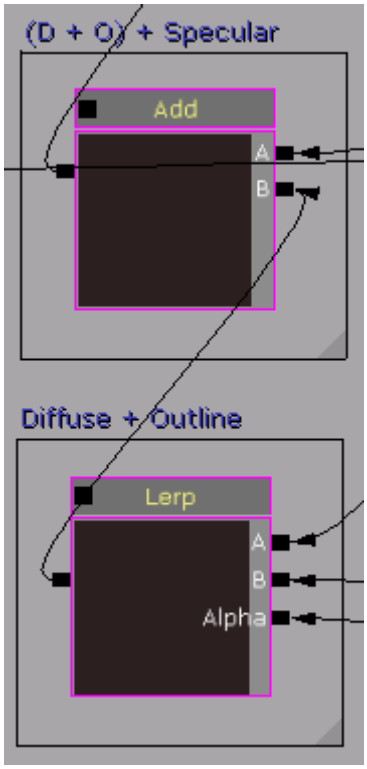


Figure 94: Adding the specular to the previously combined diffuse and fresnel.

Figure 95: (Left) The phong specular effect. (Right) The combined result.

9.2.4 – Ambient occlusion & sketch effect.

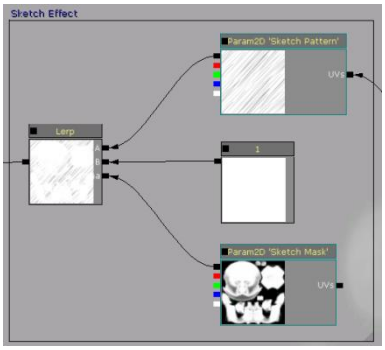


Figure 96: The sketch effect shader network.

The complete ambient occlusion effect requires three textures. The first is the diffuse texture, which already contains pre-painted ambient occlusion colour detail (in this case, painted by hand within Zbrush). The second is an ambient occlusion map, a grayscale map that contains the same pre-painted ambient occlusion detail. The third is a tiling sketch pattern.

The sketch pattern, similar to both the screentone effect in section 8.4.1 and the cloudy texture in section 9.2.1, is applied as a flat, tiling texture using the screen position process. The output is then fed through a linear interpolate using the ambient occlusion map as an alpha and a pure white constant as input B. This produces the aforementioned sketchy effect that appears only in areas that need added depth (section 9.1.2).

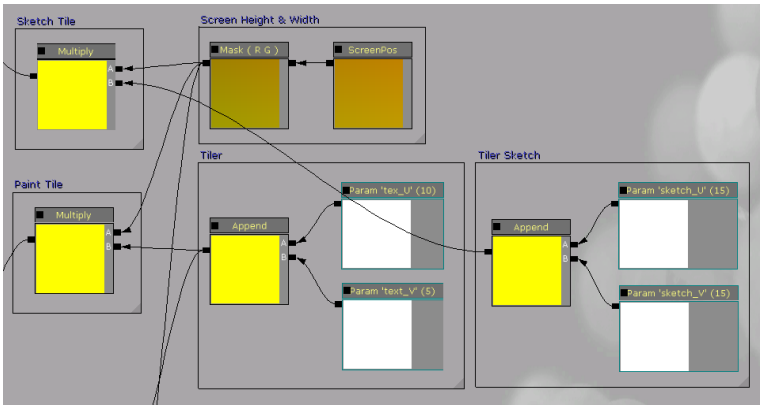


Figure 97: Extending the tiler network to accommodate the sketch effect.

This output is then multiplied over the combined diffuse, fresnel, and specular.

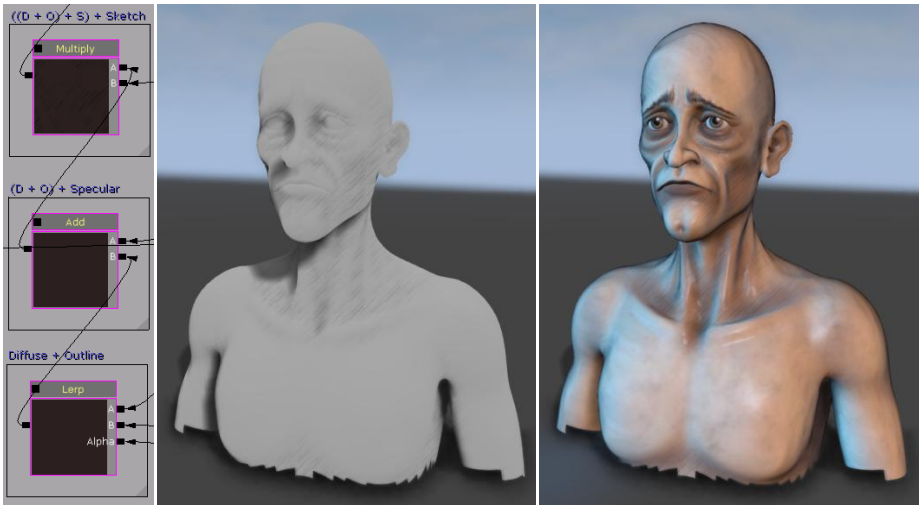


Figure 98: (Left) Applying the sketch effect to the existing surface.
(Middle) The sketch effect.
(Right) The combined effect.

Figure 99: A closeup of the sketch effect

9.2.5 – Overlay.

As noted before in section 9.1.5, part of the visual appeal of the source image is the appearance that it was drawn on dated paper, providing a rough, grungy effect to the image. This could be applied either as part of the surface shader, or as a post-process shader.

Originally, the planned method was to use a post-process shader, allowing the user to easily apply the grunge effect to the entire scene. The effect was created by taking a royalty free stock image (Buzillo-stock [pseud.], 2009), and using the screen position method to stretch it flat across the entire screen. This texture is then subsequently multiplied over the scene texture sample, allowing the scene to show through the brighter sections of the image while appearing to have folds and scratches in it.

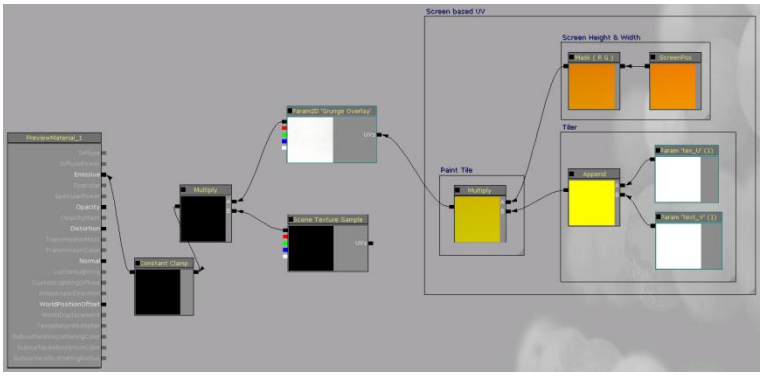


Figure 100: The initial grunge Post-Process effect network.

During the process of applying this shader however, it proved to be problematic. Should the grunge effect be applied before the depth of field (henceforth referred to as DOF) effect, the grunge effect will also be blurred. *[fig 101, top]* Placing the grunge effect after the DOF effect however, will cause the entire scene to render black. This can only be fixed by turning off anti-aliasing, leaving the final scene looking pixellated. *[fig 101, bottom inset]* The cause of this issue is unknown.

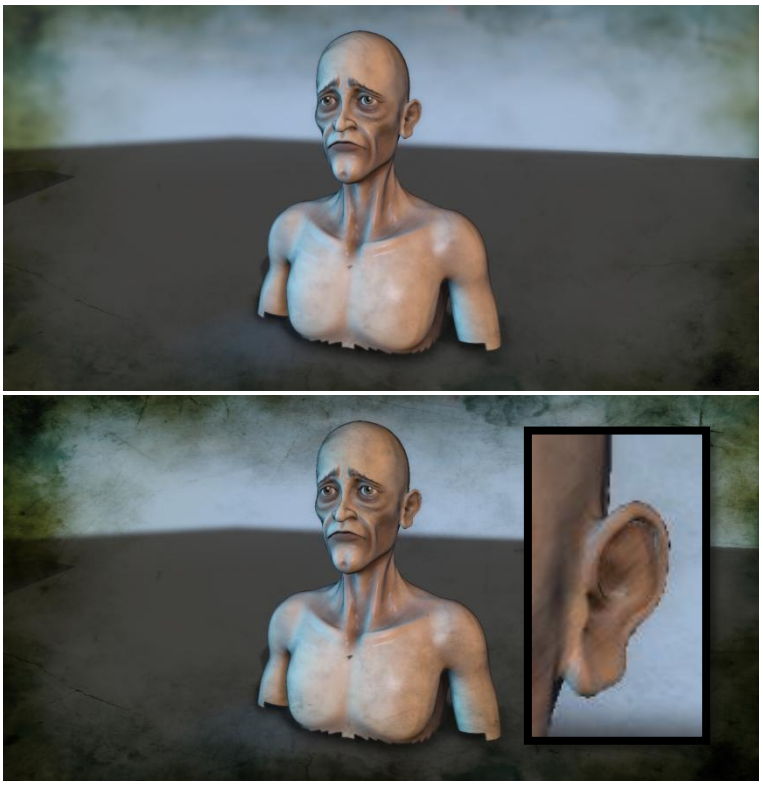


Figure 101: (Top) The grunge effect when placed before the DOF effect.
(Bottom) The grunge effect when placed after the DOF effect with anti-aliasing turned off.
(Bottom Inset) Close-up of ear with anti-aliasing turned off.

To use the grunge effect as a post-process shader will lead the end user into having to make a decision between sacrificing the DOF effect, or sacrificing anti-aliasing, both of which are considered standard features in most contemporary games.

It is for this reason that it is more suitable to apply the grunge effect as part of the surface shader, multiplying it over the combined diffuse, fresnel, specular, and sketch effect. The tiler method is extended a final time to include the settings used in the previously built post-process shader.

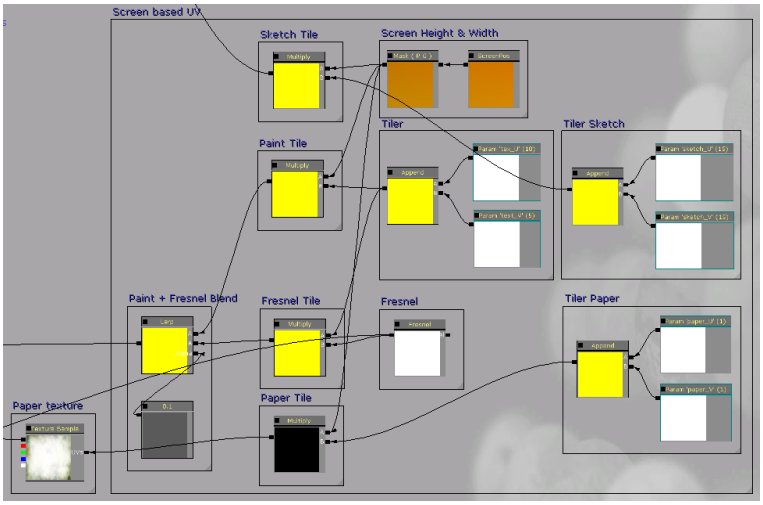


Figure 102: The final incarnation of the tiler network, accompanying the paint, sketch, and grunge effects.

While the final effect is much less noticeable than the post-process approach, this has the additional benefit of allowing the user to decide which surfaces the effect appears on (for example, excluding the sky from having the grunge effect), and to possibly apply different grunge textures onto different objects to make a scene look more varied. The increased subtlety of the effect can be beneficial, as the previous post-process method could have proven to be too distracting as the effect was very strong and noticeable.

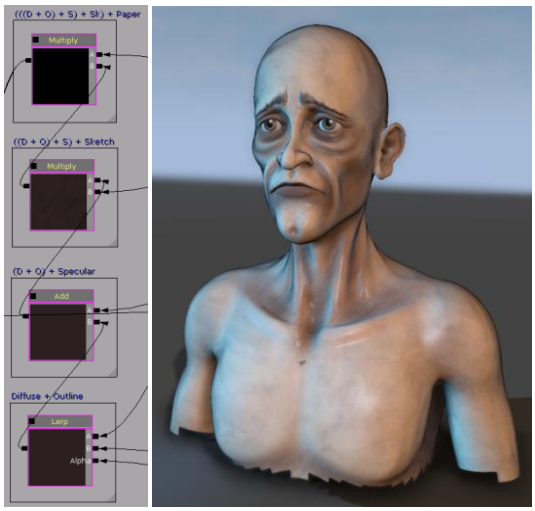


Figure 103: (Left) Combining the grunge paper effect with the existing surface.
(Right) The final result.

9.2.6 – Final Result.

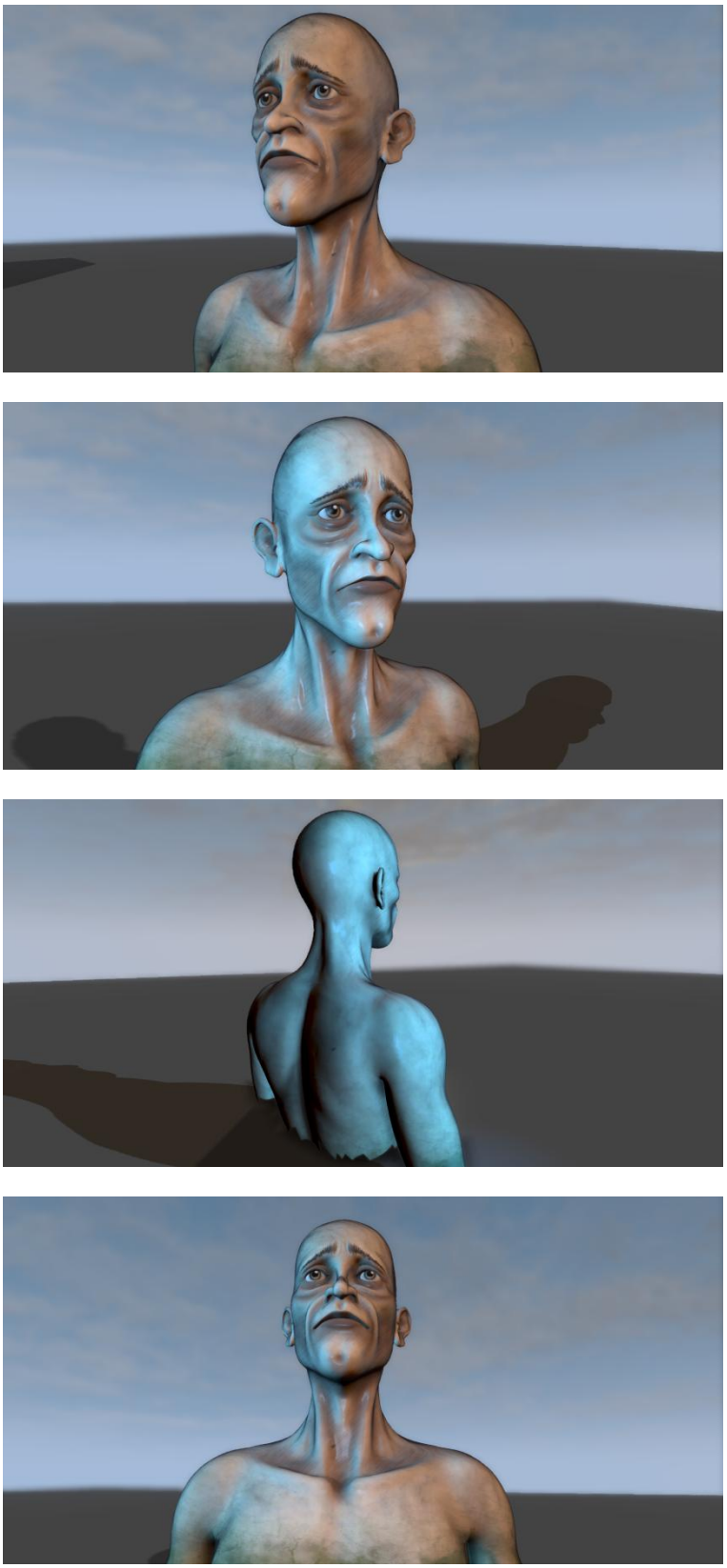


Figure 104: (Top to Bottom) Four angles of the final result.

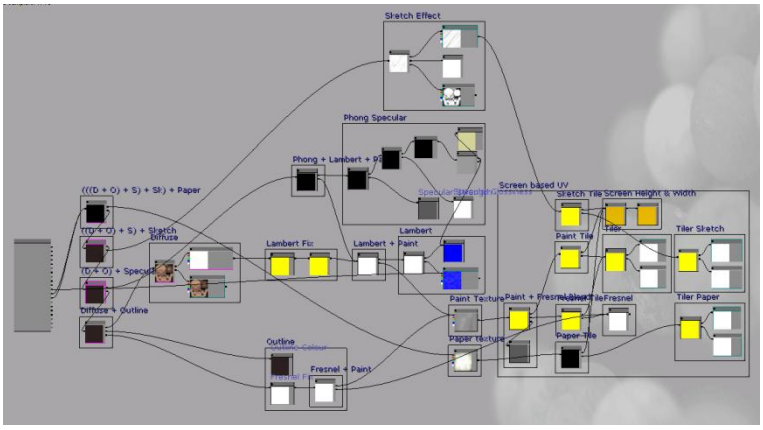


Figure 105: The final Fable 2 Concept Art style shader network.

9.3.0 – Experiment #4: Conclusion.

Due to the nature of the style, only one of the recurring obstacles mentioned in section 7.3.0 made itself known; the phong specular (section 9.2.3). As the art style is much darker and grittier, the realistic lighting produced by lightmass manages to compliment the style as opposed to breaking it, removing the need to use the workaround mentioned in sections 7.2.7 and 8.4.3. Furthermore, although the phong shader needed to be recreated from scratch, a phong shader is fortunately much simpler to create than an anisotropic specular.

The only other obstacle encountered in this experiment was with the post-process attempt detailed in section 9.2.5. This is not an issue caused by the default bias, however, but is instead a generic issue with the engine (and does not favour either photorealism or non-photorealism).

10.0.0 – Final Conclusion.

10.1.0 – The default bias and UDK.

As detailed in section 7.3.0, the hypothesis concerning the default bias was technically incorrect. Although there is evidence of the default bias existing within the Unreal Development Kit, this evidence manifested itself much earlier than anticipated. Throughout the entirety of these three experiments, only two unique examples of the default bias have been documented. Although this might seem like a small amount, these two issues are ingrained into the foundation of nearly every shader; the lighting and the shading. As such, these become recurring issues that will be encountered in most attempts at non-photorealism, as discovered in the later experiments (sections 8.0.0 and 9.0.0). Although workarounds were eventually found for these problems, they are imperfect and will require at the least minor alteration depending on the art style used.

Regardless, the fact still remains; the default bias does exist. The creation of these shaders, including the more simplistic visual novel shader (section 7.0.0), proved to be more difficult and time consuming to produce than a typical photorealistic shader. This is due to the fact that all of the features needed to produce photorealistic games are already provided as standard features within the engine, in some cases to the detriment of those who wish to produce something different.

With the default bias proven to exist, it can now be deduced that this is the true reason for the prevalence of photorealism. It has nothing to do with the popularity of photorealism, as shown by the first experiment (section 4.3.0). To the majority of consumers, photorealism is neither more nor less popular than non-photorealism; they are equally popular, and consumers wish for an equal balance of both. Instead, photo-realism's prevalence is due to it being the *easier* style. It is the default style for major game engines such as the Unreal Engine, and as such is the most supported and utilised, with many of the features required for photorealism being provided as default options. To reiterate the words of Viktor Antonov, "anybody can do photo-realism". (*Anonymous*, 2012)

10.2.0 – Extended thoughts.

10.2.1 – Additional lessons learned.

Although the initial direction of the research (section 4.0.0) proved to be unrelated to the real issue at hand (the default bias), this resulting information is far from fruitless, hence its inclusion. Although it was not directly utilised in resolving the real issue, there are some things to learn from it that are just as important.

As previously noted, non-photorealism is not unpopular. Despite the thoughts expressed and the actions taken by the companies in section 4.2.3, evidence shows that non-photorealism and photorealism are as popular as each other (section 4.3.0). Furthermore, consumers also notice a distinct lack in non-photorealistic games, and wish for both to receive equal representation. There is further unintentional evidence of this consumer behaviour through my experience as a vendor for Daz 3D, as described in section 10.2.2.

With this in mind, the act of changing a game's marketing to appear more realistic, as described in section 4.2.3, is completely unnecessary. Such changes in the presentation will not improve the game's popularity, as the majority of consumers don't favour either style. In fact, such drastic changes (such as the one proposed for *Agarest* in section 4.2.3) have the potential to make the game

less popular. Assuming the tactic does work and it attracts fans that prefer photorealism, the artwork shown in the marketing doesn't match that found in the game, a fact that will become apparent very quickly when played. As noted by some of the survey participants, this can be considered a form of false advertising (section 4.3.2).

In short, not only has this practise been proven to be ineffectual, but as a concept it is also counterproductive, potentially causing more issues than it aims to fix. Although this conclusion wasn't necessary to research the default bias, it is still an important lesson that is worth remembering; the importance of listening to the consumer can never be expressed enough.

10.2.2 – Unintentional evidence.

During the initial stages of this thesis I worked on multiple personal projects for commercial sale through Daz 3D. These included two packs of non-photorealistic shaders for *Daz Studio* (a hobbyist 3D suite). The packs are named *Visual Style* and *Manga Style*, and utilise some of the same methods shown in the visual novel (section 7.0.0) and manga (8.0.0) themed experiments.

Visual Style was originally produced as an independent project, the intention being for the product to be sold on an open store that anyone could join. After Daz 3D saw the project, however, *Visual Style* was offered a place on the official Daz 3D store (a store only for artists approved by Daz 3D). There it received advertising space alongside other popular products, including their flagship photorealistic male figure *Michael 5*. *Visual Style* far exceeded personal expectations, earning a profit of over $1,600 within the first weekend and subsequently over $3000 within the first three months (note that this is *after* Daz 3D take their half of the total earnings). The success of *Visual Style* lead to it being advertised as one of the month's "hot products", one of the other products featured being the photorealistic *Michael 5*, proving it to be just as successful as the newly released photorealistic products at the time.

Over the course of this project *Visual Style* has still continued to sell well, to the point where it was featured in Daz 3D's "Best of 2012" sale. At the time of this writing, *Visual Style* has sold over 900 copies since its first release in May 2012, earning a total profit of over $6,400. Inspired both by *Visual Style*'s success as well as the success of the manga shader experiment in section 8.0.0, a second product named *Manga Style* was created. Released in November 2012, *Manga Style* has currently earned over $3,000. Furthermore, both *Visual Style* and *Manga Style* have been chosen to be featured in *Bel*, a digital commercial magazine available from Content Paradise.

Although thematically linked to the thesis, neither of these products was intended to be a part of the thesis. Despite this, this experience as a vendor has proven itself to be parallel to the evidence produced in section 4.3.0. The fact that Visual Style was a non-photorealistic product didn't hurt its chances at success; it proved to be just as popular as the other photorealistic products at the time. If the concerns raised in sections 4.2.2 and 4.2.3 were true, this wouldn't have been the case.

10.3.0 – The future of NPR.

Much like the survey participants in section 4.3.0, I too hope to see more non-photorealistic games being produced to balance out the photorealistic games. It's not enough for there to be simply more non-photorealistic games though; the methods used to create such games need to be improved upon, just like how the methods to produce photorealism have evolved over the years.

Fortunately, there are already signs of companies choosing to pay closer attention to NPR. Since the initial research was finalised, various projects have been released that show considerable leaps in quality concerning non-photorealistic rendering.

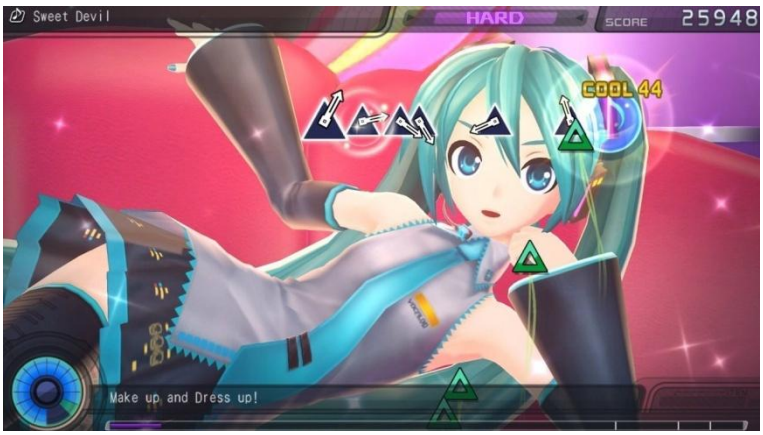


Figure 106: A screenshot of Hatsune Miku: Project DIVA F. (Lada, 2013)

In March 2013, a sequel for the Hatsune Miku: Project Diva game discussed in section 5.2.4a was released for the Playstation 3 by SEGA; Project DIVA F. *[fig 106]* Unlike its predecessor, which used realistic shading and lighting on an otherwise stylised character, Project DIVA F manages to much more accurately recreate the style of the original concept artwork through a combination of non-photorealistic textures and shaders. The game is much more vibrant and faithful to the original design, finally taking advantage of the Playstation 3's hardware to power complex non-photorealistic shaders as opposed to photorealistic ones. Furthermore, this is the first game in the Project Diva series to be officially released to western audiences, thereby acknowledging the potential consumer-base in both Europe and America.

Another mainstream project worth noting is the as-of-yet unreleased game also by SEGA, Guilty Gear Xrd. This project is particularly notable in being a 3D game that attempts to mimic the style of the original 2D games as closely as possible. This is done not only through the use of cel-shading to recreate the anime style of the original artwork, but also through the use of purposefully stuttered animation to mimic the low frames-per-second of 2D animation typical to Japan. The effect looks nigh indistinguishable to 2D animation; it is only when the camera rotates around the characters that it becomes apparent that they are 3D characters in a 3D environment. Another fact worth noting is that Guilty Gear Xrd is being developed in the Unreal Engine, the same engine used in the three shader experiments in this thesis (sections 7.0.0, 8.0.0, and 9.0.0). It would be very interesting to learn the methods they used, and to know if they too encountered similar issues to those encountered in these experiments.

While such examples from the mainstream industry are few, they are no less impressive. They also show what could potentially be a change in attitude towards non-photorealism by the mainstream industry. It may be a very long time before the mainstream industry begins producing photorealistic and non-photorealistic games in equal measure, but much like Viktor Antonov (*Anonymous*,2012) I am convinced that the future is bright for non-photorealistic games.

11.0.0 – Bibliography.

[Agarest Generations of War box art] (n.d) [image online] Retrieved from: http://spong.com/asset/307600/3/11046376

[Chrono Trigger screenshot] (n.d) [image online] Retrieved from: http://www.consoleclassix.com/snes/chrono_trigger.html

[Fable 2 conceptual art] (n.d) [image online] Retrieved from: http://www.platformnation.com/2010/06/30/fable-3-getting-an-episodic-release/fable-2/

[Mega Man Box EU] (2010) [image online] Retrieved from: http://timewarpgamer.com/features/box_art_disparity_nes.html

[Mega Man Box US] (2010) [image online] Retrieved from: http://timewarpgamer.com/features/box_art_disparity_nes.html

[Rockman Box JP] (2010) [image online] Retrieved from: http://timewarpgamer.com/features/box_art_disparity_nes.html

[Star Wars Rebel Assault screenshot] (2012) [image online] Retrieved from: http://retrogamming.blogspot.co.uk/2012/10/star-wars-rebel-assault-mega-cd.html

[Ward anisotropic distribution formulae] (n.d) [image online] Retrieved from: http://en.wikipedia.org/wiki/Specular_highlight

Anderson, C. A., Dill, K. E. (2000) Video Games and Aggressive Thoughts, Feelings, and Behavior in the Laboratory and in Life. [pdf] Retrieved from: http://www.psychology.iastate.edu/faculty/caa/abstracts/2000-2004/00AD.pdf

Anonymous (2005) "Madagascar: Bringing a New Visual Style to the Screen" - An Overview, Siggraph. Retrieved from: http://www.siggraph.org/programs/archive/reports/conference/2005/articles/madagascar/

Anonymous (2006) "Augmented perception via cartoon rendering – Reflections on a real-time video-to-cartoon system", in Computer Graphics (CG) Quarterly, Volume 40, Number 3, Pages: Unknown. Retrieved from: http://www.siggraph.org/publications/newsletter/volume-40-number-3/augmented-perception-via-cartoon-rendering-reflections-on-a-real-time-video-to-cartoon-system

Anonymous (2011) Sobel Edge Detection Post Process Effect, Unreal Developer Network (UDN). Retrieved from: http://udn.epicgames.com/Three/DevelopmentKitGemsSobelEdgeDetection.html

Anonymous (2012) "Anyone can do photorealism" says Half-Life 2 designer Viktor Antonov, Games™. Retrieved from: http://www.gamestm.co.uk/discuss/anyone-can-do-photorealism-for-next-gen-says-half-life-2-designer-viktor-antonov/

Anonymous (n.d) Amazing One-Texture Environment, Official Unreal Engine Website. Retrieved from: http://www.unrealengine.com/showcase/udk/amazing_one_texture_environment/

Antoniades, T. (2010) Ninja Theory: Story 'Most Important Part Of Game Experience', Gamasutra. Retrieved from:

http://www.gamasutra.com/view/news/121243/Ninja_Theory_Story_Most_Important_Part_Of_Game_Experience.php#.UO2n5mcXItV

Artefact [pseud.] (2010) Square Enix: "Only Macho Protagonists Work in the West", Sankaku Complex [online] Retrieved from: http://www.sankakucomplex.com/2010/05/21/square-enix-only-macho-protagonists-work-in-the-west/

Brennan, S. E. (1982) Caricature generator. [pdf] Massachusetts Institute of Technology. Retrieved from: http://dspace.mit.edu/handle/1721.1/15743

Buzillo-stock [pseud.] (2009) Grunge Textures 01, Deviant Art. Retrieved from: http://buzillo-stock.deviantart.com/art/grunge-textures-01-117072744

Carnelian [pseud.] (2000) *Suzuna Kuraki from Kao no Nai Tsuki*. [image online] Retrieved from: http://gallery.minitokyo.net/view/508208

Carter, D. (2011) *A Complete Guide: Anatomy for the Artist*. Bath: Parragon.

Cassell, J. (1998) Chess For Girls?: Feminism and Computer Fames. [pdf] Retrieved from: http://www.justinecassell.com/publications/gg_introduction.pdf

Cooper, D. (1999) Personal Thoughts on Non-Photorealistic Rendering, Siggraph. Retrieved from: http://www.siggraph.org/publications/newsletter/v33n1/contributions/Cooper.html

Dendane, A. (2012) Convert Degrees into Radians – Trigonometry Calculator, Analyze Math. Retrieved from: http://www.analyzemath.com/Calculators_2/convert_degrees_radians.html

East, T. (2010) More Amazing Mario Facts, The Official Nintendo Magazine. Retrieved from: http://www.officialnintendomagazine.co.uk/19901/features/more-amazing-mario-facts/

Flickr flame [pseud.] (2009) *Shiny Red Hair*. [image online] Retrieved from: http://www.flickr.com/photos/marcusflickrflame/3948060539/

Frick, T. (n.d) Scifi Lab, Tor Frick – 3D Artist. Retrieved from: http://www.torfrick.com/info/lab.html

Gravett, P. (2004) *Manga: Sixty Years of Japanese Comics*. London: Laurence King Publishing Ltd.

Greenberg, S. (1999) Why Non-Photorealistic Rendering?, Siggraph. Retrieved from: http://www.siggraph.org/publications/newsletter/v33n1/contributions/Greenberg.html

Heath [pseud.] (2010a) *Nier Gestalt Xbox360 screenshot.* [image online] Retrieved from: http://rpgland.com/square-enix/nier-23-screens-nier-gestalt-for-360-nier-replicant-for-ps3/#more-83938

Heath [pseud.] (2010b) *Nier Replicant PS3 screenshot.* [image online] Retrieved from: http://rpgland.com/square-enix/nier-23-screens-nier-gestalt-for-360-nier-replicant-for-ps3/#more-83938

Hofer, B. (2009) *Fable 2 screenshot.* [image online] Retrieved from: http://www.totallygn.com/2009/06/fable-2/

Inui, S. (2001) A panel from “Comic Party”, issue 1, page 20. [image online] Retrieved from: http://www.mangareader.net/1383-45292-20/comic-party/chapter-1.html

Jenni [pseud.] (2009) *The Magic of Hatsune Miku: Project Diva.* [image online] Retrieved from: http://www.siliconera.com/2009/08/18/the-magic-of-hatsune-miku-project-diva/

Kei, G. (2007) *Hatsune Miku official artwork.* [image online] Retrieved from: https://www.facebook.com/pages/Hatsune-Miku/10150149727825637

KnucklesSonic8 [pseud.] (2012) *VVVVVV screenshot.* [image online] Retrieved from: http://www.wiiloveit.com/games/vvvvvv---3DS-review

Lada, J. (2013) *Hatsune Miku: Project Diva F screenshot.* [image online] Retrieved from: http://www.technologytell.com/gaming/113142/segas-giving-us-hatsune-miku-project-diva-f/

LeClair, D. (2012) 4 Most Important Aspect Of Video Game Design That Take A Game To The Next Level [MUO Gaming], MakeUseOf. Retrieved from: http://www.makeuseof.com/tag/4-most-important-aspect-of-video-game-design-that-take-a-game-to-the-next-level-muo-gaming/

Lionhead Studios (n.d) *Fable 2 concept art “Barnum”*. [image online] Retrieved from: http://www.fanpop.com/clubs/fable/images/1588016/title/fable-2-concept-art-barnum-photo

Locke_GB7 [pseud.] (n.d) *Donkey Kong New Sprites.* [image online] Retrieved from: http://www.nes-snes-sprites.com/DonkeyKongNew.html

Loomis, A. (1943) Figure Drawing for all it’s Worth, page 172. [image online] Retrieved from: http://www.fineart.sk/photo-references/figure-drawing-all-its-worth-andrew-loomis/158

Luft, T., & Deussen, O. (2006) *Real-Time Watercolor Illustrations of Plants Using a Blurred Depth Test*. [pdf] Retrieved from: http://www.inf.uni-konstanz.de/gk/pubsys/publishedFiles/LuDe06.pdf

MakeHuman Team (2012) Can I create a commercial closed source game with models generated by MakeHuman? Retrieved from: http://www.makehuman.org/doc/faq/can_i_create_a_commercial_closed_source_game_with_models_generated_by_makehuman.html

Malone, T. W. (1980) What Makes Things Fun to Learn? A Study of Intrinsically Motivating Computer Games. [pdf] Retrieved from: http://cci.mit.edu/malone/tm%20study%20144.pdf

McMahan, A. (2003) Immersion, Engagement, and Presence: A Method for Analyzing 3-D Video Games. In: Wolf and Perron, ed. 2003. The Video Game Theory Reader. London: Routledge. Ch.3.

Meowbot [pseud.] (2006) *Screen tone example.*[image online] Retrieved from: http://en.wikipedia.org/wiki/File:Screen_tone_example.svg

Messaris, P. (1996) *Visual persuasion: The role of images in advertising*. [e book] Sage Publications, Incorporated. Retrieved from: Google Books http://books.google.co.uk/books?id=OQ5TPWYSndwC&printsec=frontcover#v=onepage&q&f=false

Mitchell, T.L. (1995) “Kid’s Stuff”: Television Cartoons as Mirrors of the American Mind. [pdf] Retrieved from: http://files.eric.ed.gov/fulltext/ED386771.pdf

Mori, M. (1970) *The Uncanny Valley.* [pdf] Retrieved from: http://www.movingimages.info/digitalmedia/wp-content/uploads/2010/06/MorUnc.pdf

Mrtheplague [pseud.] (2005a) *Ambient occlusion term rendered for an insect model.* [image online] Retrieved from: http://commons.wikimedia.org/wiki/File:Aocclude_insectambient.jpg

Mrtheplague [pseud.] (2005b) *Diffuse term rendered for an insect model.* [image online] Retrieved from: http://commons.wikimedia.org/wiki/File:Aocclude_insectdiffuse.jpg

Mrtheplague [pseud.] (2005c) *Insect model rendered with diffuse illumination and ambient occlusion.* [image online] Retrieved from: http://commons.wikimedia.org/wiki/File:Aocclude_insectcombined.jpg

Nentwich, M. (2003) *Cyberscience.* [ebook] Vienna: Austrian Academy of Sciences Press. Retrieved from: http://hw.oeaw.ac.at/3188-7inhalt?frames=yes

O’Connor, J.J., & Richardson, E.F. (n.d) Johann Heinrich Lambert, The MacTutor History of Mathematics archive. Retrieved from: http://www-history.mcs.st-andrews.ac.uk/Biographies/Lambert.html

O’Hare, J. (2012) Anisotropic Highlight Shader, Unify Community. Retrieved from: http://wiki.unity3d.com/index.php?title=Anisotropic_Highlight_Shader

Price, J. (2006) Opinion: Is Photorealism In Games The Right Direction?, Gamasutra. Retrieved from: http://www.gamasutra.com/view/news/102441/Opinion_Is_Photorealism_In_Games_The_Right_Direction.php

Prince Ludwig [pseud.] (2012) *New Super Mario Bros U.* [image online] Retrieved from: http://www.mariowiki.com/File:Mario_-_New_Super_Mario_Bros_U.png

Rautek, P., Bruckner, S., Gröller, E. (2008) “Illustrative Visualization – New Technology or Useless Tautology?”, in Computer graphics (CG) Quarterly, Volume 40, Number 3, Pages: Unknown. Retrieved from: http://www.siggraph.org/publications/newsletter/volume-42-number-3/illustrative-visualization-2013-new-technology-or-useless-tautology

Rossiter, J.R. (1982) “Visual Imagery: Applications to Advertising”, in Advances in Consumer Research Volume 09, eds. Andrew Mitchell, Advances in Consumer Research Volume 09: Association for Consumer Research, Pages: 101-106. Retrieved from: http://www.acrwebsite.org/search/view-conference-proceedings.aspx?Id=5909

Shuman, S. (2012) “Kara” is Not Quantic Dream’s Next Game (But You May Wish it Was). Retrieved from: http://blog.us.playstation.com/2012/03/07/kara-is-not-quantic-dreams-next-game-but-you-may-wish-it-was/

Spencer [pseud.] (2009) Different Region, Different Star Ocean: The Last Hope Interface. Retrieved from: http://www.siliconera.com/2009/02/06/different-region-different-star-ocean-the-last-hope-interface/

Spencer [pseudo.] (2010) *Here's Hatsune Miku Rendered On A Playstation 3.* [image online] Retrieved from: http://www.siliconera.com/2010/04/23/heres-hatsune-miku-rendered-on-a-playstation-3/

Tezuka, O. (1952) *The cover for Atro Boy volume 1 and 2 compilation by Dark Horse Comics.* [image online] Retrieved from: http://en.wikipedia.org/wiki/File:AstroBoyVolume1.jpg

The Society for the Study of Manga Techniques (2003) *HOW TO DRAW MANGA Volume 1: Compiling Characters*. Tokyo: Graphic-sha Publishing Co., Ltd.

Tolentino, J (2009) Record of Box Art Failure: This was nearly your Agarest War. Retrieved from: http://www.japanator.com/record-of-box-art-failure-this-was-nearly-your-agarest-war-11179.phtml

VampirGoth [pseud.] (2012) *Lady Gaga Caricature.* [image online] Retrieved from: http://vampirgoth.deviantart.com/art/Lady-Gaga-Caricature-293023036

Wages, R., Grünvogel, S. M., Grützmacher, B. (2004) How Realistic is Realism? Considerations on the Aesthetics of Computer Games. [pdf] Retrieved from: http://nomadslab.org/wages/content/wages_icec2004_realism.pdf

Wang, J. (2012) "A New Dawn" DirectX 11 Demo Available For Download, GeForce. Retrieved from: http://www.geforce.com/whats-new/articles/a-new-dawn-directx-11-demo-available-for-download

Woolfson, A. (2006) Why Most Manga Books Are Small and Black-and-White, Yaoi 911. Retrieved from: http://www.yaoi911.com/why-most-manga-books-are-small-and-black-and-white/

Wright, D. (n.d) Custom Lighting, Unreal Developer Network (UDN). Retrieved from: http://udn.epicgames.com/Three/CustomLighting.html

XSEED Games [pseud.] (2010) Response to message from fan Zak McShane, Facebook. Retrieved from: https://www.facebook.com/XSEEDGames/posts/410775498353

Yagami, C. (2002a) *A panel from "Fall in Love Like a Comic!", issue 9, page 10*. [image online] Retrieved from: http://www.mangareader.net/1260-43206-10/fall-in-love-like-a-comic/chapter-9.html

Yagami, C. (2002b) *A panel from "Fall in Love Like a Comic!", issue 1, page 26*. [image online] Retrieved from: http://www.mangareader.net/1260-43198-26/fall-in-love-like-a-comic/chapter-1.html

Yagami, C. (2002c) *A panel from "Fall in Love Like a Comic!", issue 9, page 5*. [image online] Retrieved from: http://www.mangareader.net/1260-43206-5/fall-in-love-like-a-comic/chapter-9.html

Yoshinori, S. (1999) Manga and Non-Photorealistic Rendering, Siggraph. Retrieved from: http://www.siggraph.org/publications/newsletter/v33n1/contributions/Yoshinori.html