Damien Lappa

Photorealistic Texturing for Modern Video Games

Bachelor's thesis
Game Design

2017

South-Eastern Finland
University of Applied Sciences

Author (authors)	Degree	Time
Damien Lappa	Bachelor of Culture and Arts	November 2017

Thesis title
Photorealistic Texturing for Modern Video Games — 45 pages

Commissioned by

Ringtail Studios OÜ, Xamk South-Eastern Finland University of Applied Sciences

Supervisor

Marko Siitonen, Lecturer

Abstract

Simulating realism has become a standard for many games in the industry. While real-time rendering requires considerable rendering resources, texturing defines the physical parameters of the surfaces with a lower computer power.

The objective of this thesis was to study the evolution of Texture Mapping and define a workflow for approaching a photorealism with modern instruments for video game production. All the textures were created with the usage of Agisoft Photoscan, Substance Designer & Paintrer, Abode Photoshop and Pixologic Zbrush.

With the aid of both the theory and practical approaches, this thesis explores the questions of how the textures are used and which applications can help to build them for a better result. Each workflow is introduced with the main points of their purposes as the author's suggestion, which can be used as a guideline for many companies, including Ringtail Studios OÜ.

In conclusion, the thesis summarizes the outcome of the textures and their workflow. The results are successfully established by the author with attendance to introduce methods for the material production.

Keywords

real-time rendering, 3D texturing, computer graphics, physically-based shading

CONTENTS

LIST OF CONCEPTS

2D	2-Dimensional.
3D	3-Dimensional.
3D Asset	an object (also called model or mesh) that is meant to appear in a video game.
Bitmap	an image in digital form, where for each pixel element is assigned one bit of information. Also can be referred as Map.
Game Engine	a software framework designed for the creation and development of video games.
Materials	are definitions of how a surface should be rendered, including references to textures used, tiling information, color tints and more.
Polygon	a surface with 3 or more side, defined by vertices, for visualization in a three-dimensional graph.
Procedural Texture	a computer-generated image created using an algorithm intended to create a realistic surface or volumetric representation of natural elements in Texture Mapping.
Real-time rendering	production of visualization in real-time.
Shader	small scripts that contain the mathematical calculations and algorithms for calculating the material.
Texture	an image for defining the surfaces' color, reflection or relief information.
Vertex	a single point whose sole property is its position in 3D space.

1 INTRODUCTION

Video games have existed for a long time enough to become a part of popular culture of the modern age. Nobody could have predicted that in less than five decades the games would grow from a pure form of shapes of the pixels to the global industry with the expected of $108.9 billion in revenues in 2017 (McDonald 2017).

Since the first 3D video games were introduced, the real-time rendering has stepped closer to the realism more than ever before. Despite the fact that the purpose of the games is to bring a simulation of the experience to its player, the environmental aesthetics have become a massive part in a game production, and with its visuals, it has become a part of that experience that makes a modern game even more memorable.

The primary focus of this thesis is to achieve to produce the believable game textures and to present the workflows used by the author. The thesis covers the basis of the evolution of Texture Mapping in video games and concentrates on the aspects of the 2D imagery usage in a three-dimensional space. The contents are intended for the artists who already know the basics of 3D modeling and UV Mapping and does not present any mathematics or code.

For demonstration purposes, all the textures were produced from scratch by the author and were rendered for the final presentation in real-time using both Toolbag 3, developed by Marmoset, and a built-in renderer in Substance Designer, developed by Allegorithmic.

All methods were based on the authors' own experience combined with research of creating modern environment material workflow using the latest updates of the various software and mixing them for the faster approach including the renders to apply the theory in practice.

2 EVOLUTION OF TEXTURE MAPPING IN 3D GAMES

Visual graphics are an essential part of video games nowadays. Designed to set the illusion of reality to the human eye of the virtual world, the prime indicator of the technology that powers it is on the computer itself.

Computer graphic is a discipline of producing an image with computer calculations. The process includes the manipulation of visual and geometric information and though it has a short existence it has made a dramatic advance in different mediums, leaning towards more and more realism.

The technology provided the possibility to create an artificial simulation, but due to the intensive use of computer resources for the real-time rendering, it took time for the gaming industry to find a way to represent a reality on the screen via Texture Maps.

2.1 Foundation of Computer Graphics

Since the electricity became an unpreventable part of the human's life, it was a matter of time when the technology would make its way to become an object of many experiments leading to visualization graphics in video games.

Based on the article (Jones 2017), in the middle of 19th century the cathode rays, or electric beams, were a subject of research by many physicists and found its usage in television and computer screen to produce both vector and raster graphics. The technology was based on the flow of electrons in a vacuum tube moving from a negatively charged electrode at one end to a positively charged electrode in another transverse voltage. The differences between the electrodes cause the material to glow, bringing the pixels of the image to the screens and were called cathode ray tubes (CRT). Invented in 1897, Oscillates were the first to use cathode ray tubes as the two-dimensional displays. (Wolf 2008, 9-12.)

Shortly after the World War II, in 1949, the Whirlwind, the very first mainframe computer to use a CRT as a graphic display, was developed at the

Massachusetts Institute of Technology. In 1951, the system was demonstrated to the publicity with its capabilities to display real-time video, text, and graphics on a big oscilloscope screen, which has brought a high value for military purposes. (Wolf 2008, 17.)

Early Computer Graphics and systems continued to distribute a high interest for different experimentations for the following decade. Together with CRT technologies, they have found its predecessors not only in scientific studies, but also in artforms. In 1960, William Fetter was first to introduce the term Computer Graphics while attempting to shape an efficient process of a layout inside Boeing's airplane cockpits (Shklyar 2004). As can be seen in Figure 1, his result was a computer generated orthographic view of the human form which founded a revolutionary step forward in design methods of that period.

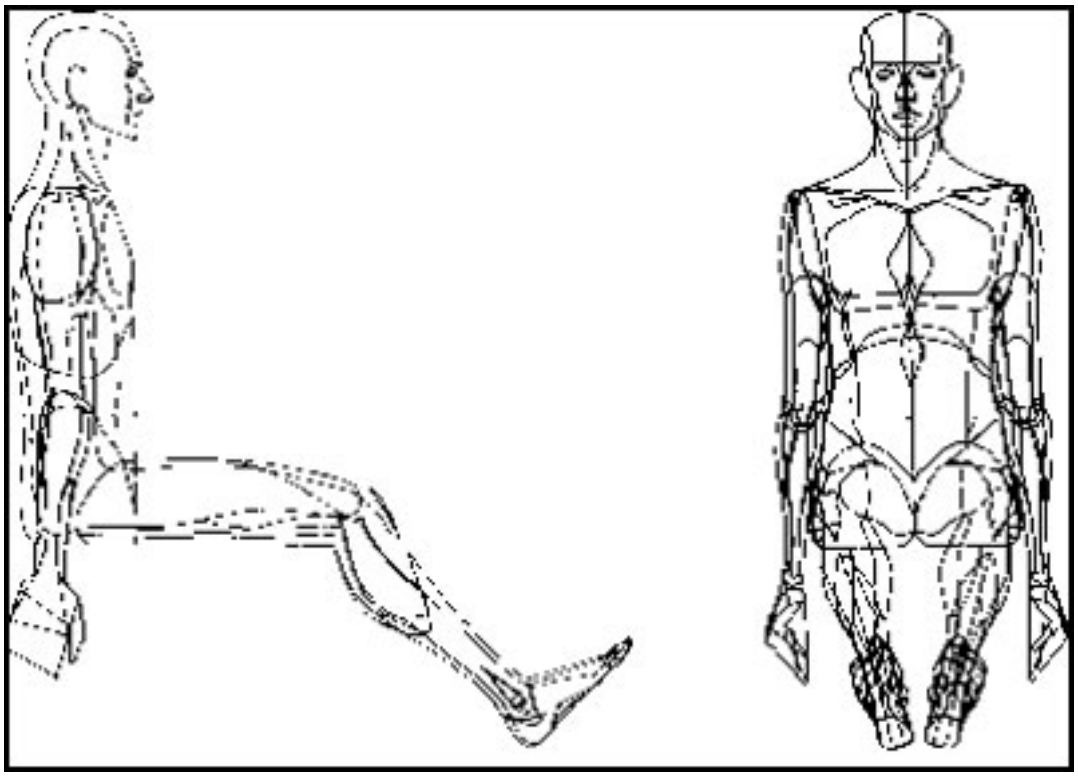

Figure 1. Computer Graphics in the 1960's (Fetter, n.d.)

At the University of Utah, one of Fetter's contemporaries, Ivan Sutherland made a revolution in the CG industry, bringing an interactive communicative machine system, called Sketchpad. It featured a constraint-based drawing and hierarchical modelling utilized with lightpen for interaction on top of oscilloscope screen (Wolf

2008, 17-18). With his success, Sutherland formulated the ideas of using primitives (such as lines, arcs, vectors) and their manipulations inside of the computer system making him be considered as the founder of the computer graphics (Shklyar 2004).

According to the article (Chapman 2015), Ivan Sutherland, together with his partner Dave Evans, later, in 1968, formed the very first computer graphic related company, called Evans & Sutherland. The researchers of the group made a considerable contribution to 3D graphics industry with developing the hidden-surface algorithm for rendering and display technology which is fundamental for the modern real-time 3D rendering in video games even nowadays.

2.2 Defining Texture Mapping

During the early 1970s computers became more capable of producing 3D geometry with much more shaped polygons. During the research, developers faced an issue of computer memory performance, where, since the first renderers were able to present only the flat shading model, the only way to increase the smoothness of geometry with more details was to add more polygons. (Wolf 2008, 48.)

A French computer scientist, Henri Gouraud, found a solution to one of the issues. He utilized a unique shading by interpolating the normal of the vertexes. Gouraud Shading resulted in a smooth surface that takes only a small amount of the system’s power without increasing the number of the polygons. Later, the shading model technique was expanded by Phong Bui-Tuong. He introduced a calculation of the normal vectors for each pixel. Phong resulted in an incredibly smooth surface with definite highlights, but it used much longer time to render compared to the previous shading model. Both shading techniques can be seen in Figure 2 in comparison to the flat geometry. (Shklyar 2004.)

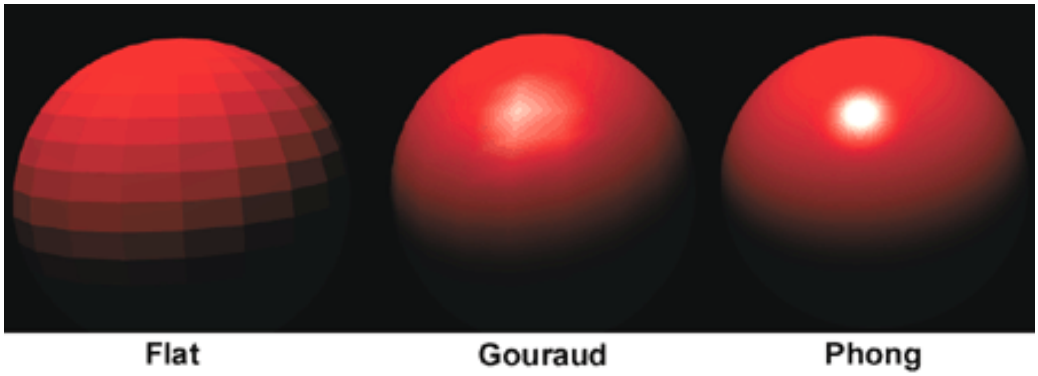

Figure 2. Types of Shading (PCMAG, n.d.)

According to the article (Blinn & Newell 1976), the fundamental work in Texture Mapping is attributed to Edwin Catmull. In 1974 Catmull recognized an easy way to achieve realism in 3D geometry by producing a picture on surface polygons with a minimal effort of the computer systems. He developed an algorithm for rendering images and was the first to demonstrate the Mapping of a texture pattern onto arbitrary planar and cylindrical surfaces. The algorithms apply two-dimensional pictures like a skin to the UV grid of projected polygon surfaces, adding the rasterized details to a computer-generated scene.

James Blinn and Martin Newell (1976) later refined the Texture Mapping even further. By combining Catmull's Texture Mapping technique with 2D picture elements and Phong's reflection shading to the teapot mesh, they resulted in an image of a highly glazed by that time with a fully textured 3D geometry object, which can be seen in Figure 3.

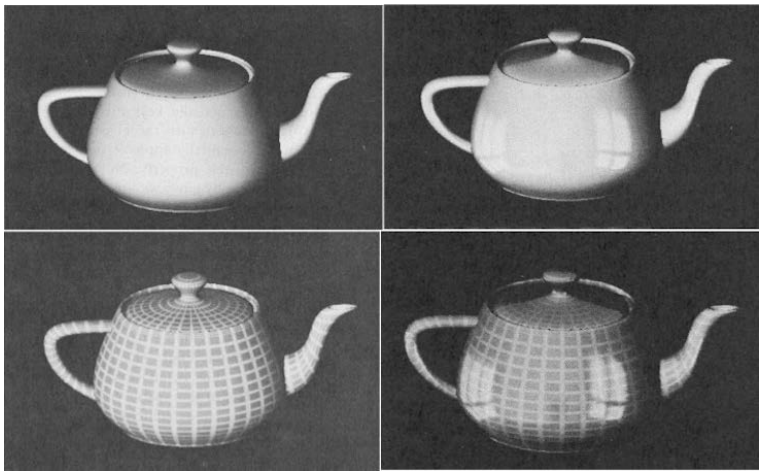

Figure 3. Texture and Reflection (Blinn & Newell 1976)

During 1978, James Blinn also achieved in building an illusion of some very sophisticated geometry without the increased number of polygons - Bump Mapping. With the usage of black and white colors, the texture defines the information of which pixel will have a relief on arbitrary polygonal models in real time. Bump Displacement function later extended the Mapping even further. This technique is based on similar Bump Mapping function, except the actual vertexes are transformed with an exact position. The process helped to build the silhouette of geometry for extra details but led to a severe load of processor power and an additional memory usage. (Policarpo et al. 2005.)

2.3 3D Game Texture Mapping Evolution

Physically based approaches to rendering started to be seriously considered by graphics researchers in the 1980s. One of the techniques of such production is called ray-tracing (Whitted, n.d.). The algorithm takes inspiration from the reality, which simulates the light bouncing in the scene, producing the reflection based on the Texture Mapping parameters. Ray-tracing delivered realistic graphics and was actively used in movie productions, but at a higher cost for the calculations even nowadays.

Real-time 3D graphics in the late 1980s and early 1990s were extremely computationally intensive. The first implementation of 3D with a full surface texture rendering was introduced in Ultima Underworld. The game had an environment that could be viewed from arbitrary oblique angles and was revolutionary during that period, but required a significant amount of power to build a scene in real-time. (Williams 2017.)

Influenced by the development of Ultima Underworld, a small by that time game company named id Software released Catacombs-3D in 1991 with the 3D approach of Texture Mapping and later expanded it in Wolfenstein 3D, in 1992. The games resulted in a significant reliance on 2D imagery to supplement the illusion of 3D forms called ray-casting. (Kusher 2003, 91-92, 97-98.)

Ray-casting involved the projection of a cone of rays from the player's position on a 2D Map. The levels were built on a simple square grid with the walls being entirely Texture Mapped, as can be seen in Figure 4. Combined with simple level geometry game were performant for the machines that time. (Pernady 1996.)

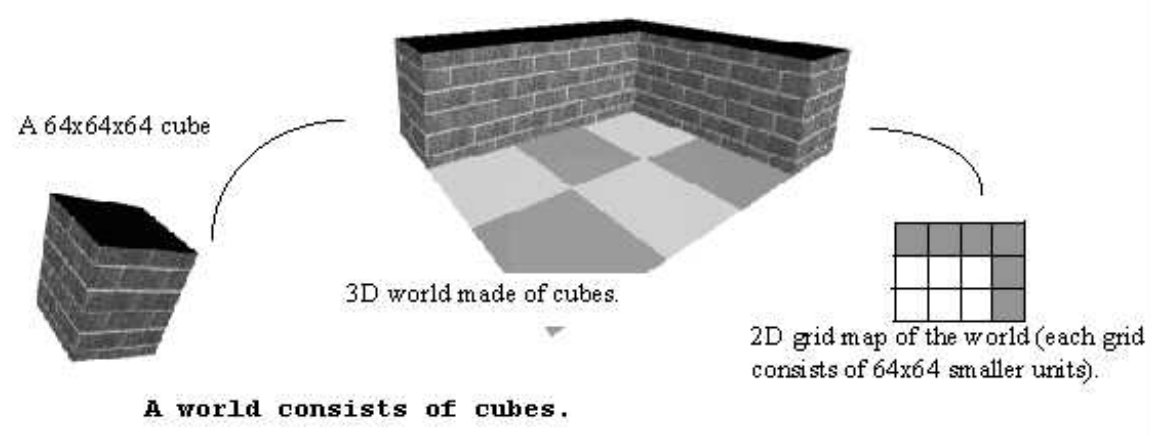

Figure 4. Ray Casting Example (Permady 1996)

Later, id Software has increased the capability of Texture Mapping by implementing Binary Space Partitioning (BSP) algorithm with the ray-casting to their new game - DOOM. Limited to a vertical axis of rotation in a 2D space, the engine of the game allowed to divide the sectors of a level Map to simulate in real-time the variety of room heights with a full texturing of floors, ceilings, and walls. (Kusher 2003, 142-143.)

The rapid evolution of game technologies in the mid to late 1990s significantly improved the range of application and opened the doors for the higher level of detail. Real-time 3D rendering has established the definition of a Texture Mapping of the surfaces, which contained the essential color information with Gouraud's shading, but without any other physical approaches yet.

In 1996, the Texture Mapping saw a breakthrough with the release of Quake. Developed by id Software, the game was not only the first to introduce complex 3D scenes with characters and environments but also introduced illumination Maps or Light Maps to capture lighting effects in video games. (Williams 2017.)

Static lighting information started to be captured and stored in a Texture Map covering all polygons of the level, at a much lower resolution than the Texture Maps themselves. With the release of graphics amplifier chipsets, id Software distributed an updated version of Quake that could render the Light Maps directly on the textures, using the multi-texturing method with alpha-blending, which would see similar reuse power in many other games (Shahrani 2006).

By 1998, the rise of 3D acceleration allowed the game developers to take advantage of the power to construct smooth and detailed worlds without the significant compromise. 3D Accelerator cards permitted to unlock the real-time rendering limits of 3D gaming with both better performance and quality, and it was a matter of time when games would become entirely dependent on them.

Developed by Epic MegaGames, Digital Extremes, and Legend Entertainment, Unreal was the game that took full advantage of the hardware and quickly became associated with cutting-edge graphics. With diffuse and lighting textures, Unreal is one of the first games to utilize the Detail Texture Mapping technique (see Figure 16, 29). It allows enhancing the surfaces of objects with a second texture that shows material detail to build a believability of the high-resolution of the Maps (Shahrani 2006).

By the early 2000s, the Texture Mapping has evolved from simple 2D sprites to a real polygonal surface with the multiple textures applied on top of it. The games pushed realism to the new heights and attracted the new generation of gamers to become a part of that experience.

2.4 Advancing Realism in 3D Games

3D game visuals before the early 2000s relied heavily on Color Maps and the baked lights alone. Between the 2000s and 2010s, the evolution in game graphics continued to proceed as each new generation of hardware allowed developers to provide more resources for building more realistic visuals.

With the release of Quake 3 Arena, id software refined the usage of Shader scripts, which significantly increased the capabilities of Texture Mapping in games (Jaquays & Hook 1999). With the ability of Shaders, artists and programmers now could adjust the properties of a surface with multiple Texture Maps to add a physical appearance in real-time rendering, which had already been used in 3D movie production by that time. With the opened horizons, new graphics standards required to implement new plans for developing the games. Now that it was plausible to transfer the materials attributes, the Color Map was apparently not enough.

With advances in hardware, developers found an opportunity to produce multiple textures for simulating ray-tracing rendering effect in real-time rendering. Bump Maps were ones of the earliest additions that significantly refined game visuals as they imitated subtle surface irregularities and gave the illusion of height and depth on a flat surface. Bump Mapping found its early usage in Jurassic Park: Trespasser. Later, from a simple grayscale image, Bump Mapping was replaced by Normal Maps (Figure 11, 23), which provided sharper detail by converting the surface of a high polygon 3D model to a 2D Map and overlaying it on a low polygon model. This technique added a more visual feature without adding more computation intensive polygons and resulted in higher performance. (Williams 2017, 228.)

Shaders are also used to simulate the effects of light on models and the game environment through Specular Maps, which controlled the reflective properties of a surface, allowing highlights that automatically reacted to different light intensities. (Williams 2017, 229.)

By the end of the 2000s, the game industry had refined the principles of physically based shading. Choosing the right building materials had become essential for the believability of games around the playable characters. Texture Maps are there to define the strength of objects and structures, and the gaming industry is still perusing the abilities to produce the realism in real-time rendering.

3 PHYSICALLY BASED TEXTURE TYPES

Physically-based shading is a technique that, if thoroughly defined, is the closest approach of the reality in real-time rendering nowadays. The term is bandied around the primary fundamentals of the behavior of light and matters and can be referred to as Physically Based Rendering (PBR) or Physically Based Shading (PBS) (McDermott, n.d. b). According to McDermott (n.d. a), the ultimate goal of the textures is to describe the individual parameters of how the surface of a 3D object is reacted with the connection of the light within the environments.

Since shading capabilities have advanced enough, some of the old approximations are evaluated to the new means of producing a textured art for photorealistic looks. Though the Shaders are written by programmers, artists are there to achieve the best results with the creation of inputs & outputs for the further material manipulations. (Ahearn 2008, 95.)

It is essential to understand the fundamentals of light and shadow to build a physically correct texture property. Nowadays, reflections are based on the combination of the imagery, where every parameter of the Map relies on the basics of light and its perception of the human eye. In this chapter, the author describes the most common types of textures.

3.1 Base Color Map

Light is a produced transverse electromagnetic wave that consists of fluctuations of electric and magnetic fields in nature. The retina of the human eye is sensitive only to a limited radiation at wavelengths that range between 380 nanometers and 740 nanometers, which are visible in the RGB spectrum of colors. (Adobe, n.d.)

The surface of an object reflects some the mixture of wavelengths and absorbs all the others. Human’s brain perceives only the mirrored shade via the electrochemical signal in red, green and blue hue range (Pantone, n.d.). As shown in Figure 5, the surface reflects the specific wavelengths and absorbing all

the rest, and it appears as a value that is revealed. If the object appears white, it indicates all wavelength, while black consumes them all.

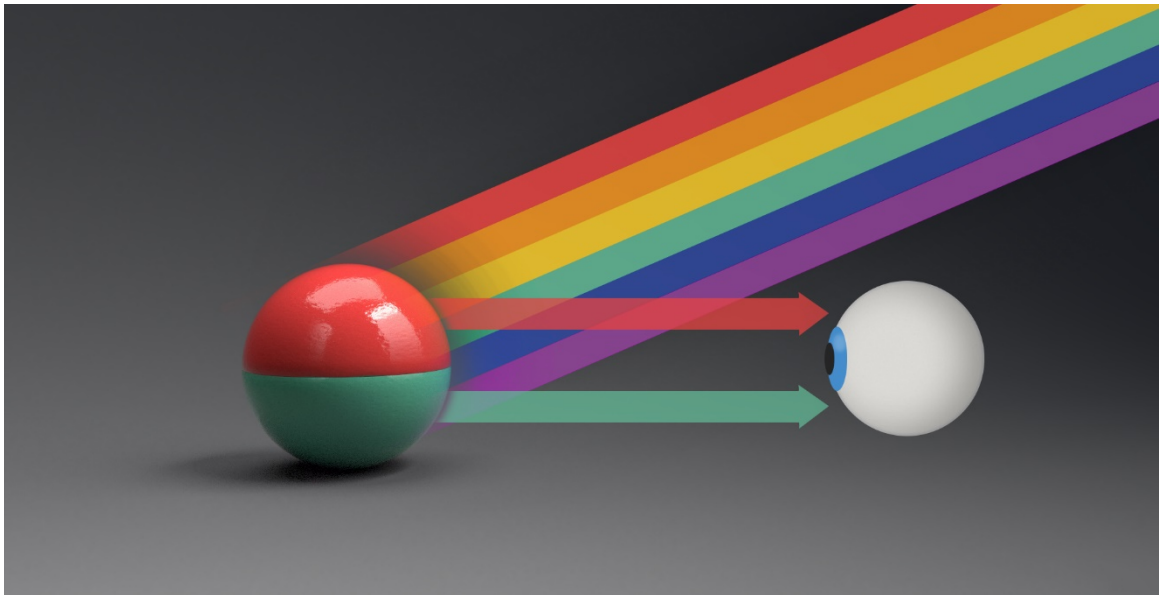

Figure 5. Color perception (Lappa 2017)

The very first of Texture Maps to describe this effect in video games was a Diffuse texture, also called Color and Albedo Map. According to Ahearn (2008, 101), the initial part of this Map was to tell a player what material appears before him by a single image on top of the surface.

With the introduction of the physical properties of the Maps, the process of this texture became much more straightforward. Base Color Map became an initial part of describing an appearance in PBR, playing a critical role in the way how the lighting is diffused. The texture defines the tint of a material without any lighting information because those effects come from the contribution of other Map types, such as Specular Maps. (McDermott, n.d. b.)

Figure 6 illustrates the difference, where Color Map conveys the subtle details and leans toward the flat image of the surface, where it excludes the directional lights and ambient occlusion. If the additional lighting information is baked into the texture, the appearance will become incorrect in certain different lighting conditions.

Figure 6. Base Color example (Lappa 2017)

Base Color defines merely the overall tint of the material, and its value is varied between 0, which equals to black in hue, saturation & lightness (HSL) and 1, which has a value of 255 of every color in sRGB space and equals to white. Though some colors in real life are preserved in our brain as a constant value, nothing in real life is entirely black or white (Pettit 2015). Such appearance is achieved because of the lighting conditions under which they are viewed, and according to the table, the black charcoal equals to 0.2 value and white snow is measured as 0.81 in a total intensity. (Epic Games, n.d. c.)

3.2 Reflection Maps

In real-time rendering, all the objects are evident because of the reflected light from different illuminates. When light hits the surface, two things happen - diffusion and specular reflection.

While the diffusion scatters light in many directions and leans some back towards the viewer (Figure 5), it must first penetrate the surface. Energy Conservation plays a vital role in delivering that function in physically-based rendering. With the higher level the intensity of the reflection, the Shader keeps the strength of the

diffuse scattering lower, leaving the surface with the same brightness value. From the perspective of an artist, this aspect allows focussing only on one amount of the roughness of the material, rather than always adjusting the brightness of the diffusion as well. (McDermott, n.d. a.)

The word specular, translated from Latin “mirror,” is used to describe a ray of a radiant from the surface that is headed in a constant direction on the opposing side of the surface, as it can be seen in Figure 7. However, most objects do not have such high smoothness, and therefore the direction of reflection will vary depending on how much the surface is incoherent at different angles. (Russel 2015.)

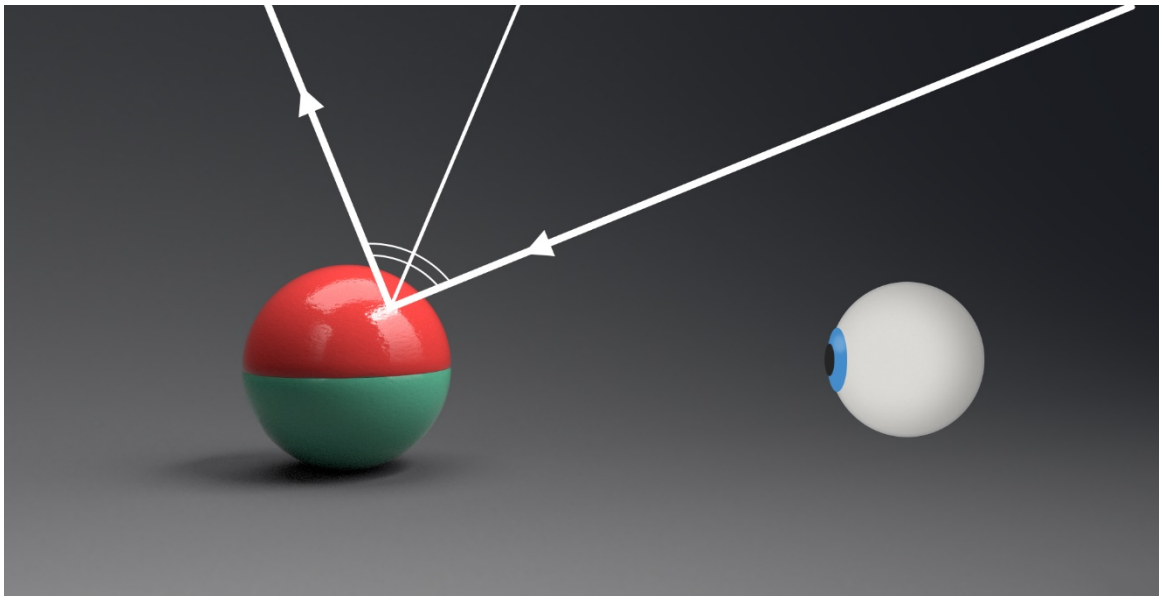

Figure 7. Light reflection (Lappa 2017)

As stated by McDermott (n.d. a) and can be seen in both Figure 8 and Figure 9, with the same light conditions the surfaces might appeal differently. While the smoothness of the surface is expressed in a more intense mirror image, thus causing a sense of saturation and brightness of the illumination, the coarser surface has a much dimmer flare. The measure of smoothness is often referred as Gloss or Roughness Maps in Physically-Based Shading.

Following the reality, it is also important to consider the Fresnel Effect. According to Unity User Manual (2017c), the effect itself manifests the material at high

viewing angles, which is expressed in a high ability to reflect light. This impression can be noticed in Figure 8 and Figure 9 as well.

When creating materials for PBR, it is also essential to consider the structure of the surface. There are two types of the properties how the light reflects from an element: insulators or dielectrics and electrical conductors, which are also known as metals. According to Russel (2015), metallic materials usually have a reflectivity of up to 60-90%, while non-metals have a much lower range, about 0-20%. Figure 8 and Figure 9 also illustrates the comparison to insulators, where electrical conductors absorb rather than scatter lighting rays that penetrate the surface, without any diffuse light. The measure of smoothness is often referred as Specular or Metalness Maps in Physically-Based Shading.

3.2.1 Specular & Glossiness Workflow

Specular & Glossiness workflow is defined through a set of Maps, which are combined as a set of textures in Physically Based Shading. As it can be seen in Figure 8, the data of the values of each surface have different results.

Gloss textures define the blurriness or sharpness of the reflections across a texture's surface with the microscopic roughness of a material’s surface. The brighter the gloss texture, the more apparent it will appear, where the darker value represents, the rougher surface reflection. (Wilson 2015.)

Specular Textures are used to define metal and non-metal areas on a surface. They offer more control over the specular intensity and allow greater flexibility when trying to reproduce specific complex materials, but require an intense understanding of physical material properties to get the right values. (Wilson 2015.)

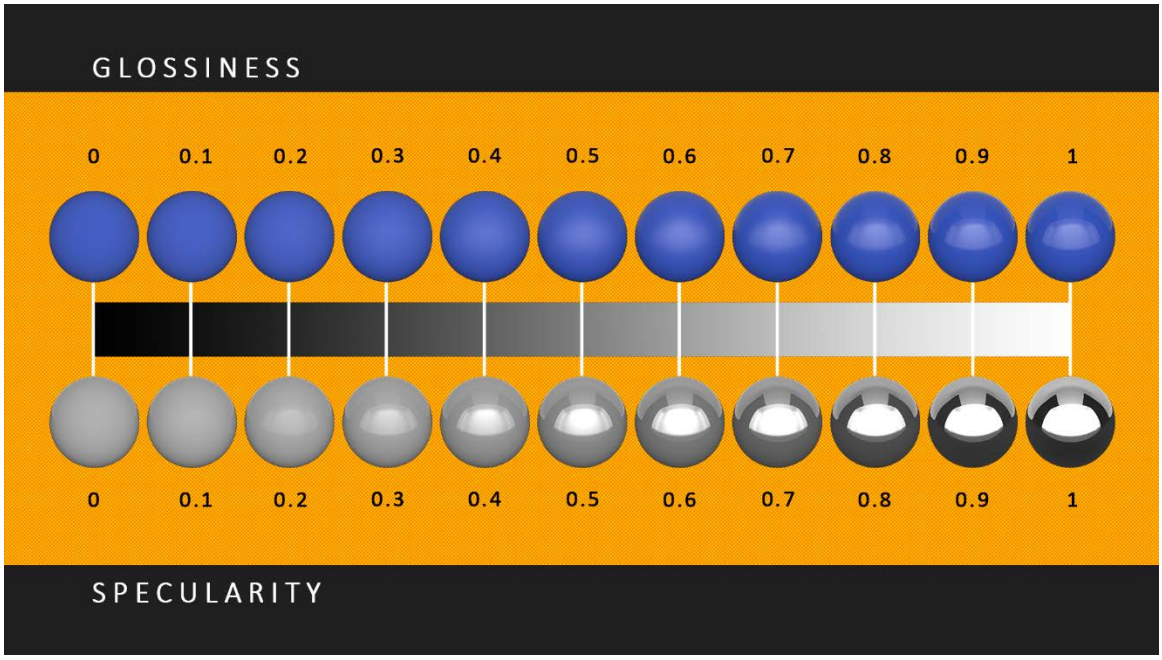

Figure 8. Glossiness (up) and Specularity (down) difference in values (Lappa 2017)

Pristine bare metallic surfaces need to have a pure black Color Texture, where its color is placed in Specular Map. It is also important to consider, that metal that is weathered, oxidized or painted needs to be treated as a dielectric, and if the rule is not followed, the reflection of the material will be miscalculated. (McDermott, n.d. b.)

Overall, Specular/Glossiness can provide excellent control over the dielectric surfaces if handled correctly, but it utilizes a bit more of the texture memory with extra RGB channels in Specular Map.

3.2.2 Metalness & Roughness Workflow

An alternative way to produce a reflectivity in PBR is Metalness & Roughness pipeline. The difference between Specular & Glossiness is in marking materials as metallic or non-metallic, and definition of the surface smoothness. However, as can be seen in Figure 9, the overall result of the reflection remains the same.

When using a Metalness Map, insulative surface's values are set as a constant to 0, which equals black, while metals are marked in 1.0 in a greyscale color space which equals white. With this workflow, artists have a more accessible approach

to the surface reflection, where the whole control is set to the Roughness Map. (McDermott, n.d. b.)

According to McDermott (n.d. b), this method allows the Albedo Map to work with metals regularly like with dielectrics in Specular & Glossiness pipeline. Though the roughness is also set to the black & whites, the reflection is controlled by different values. The full result of the difference rendered by the Author can be seen in Figure 9.

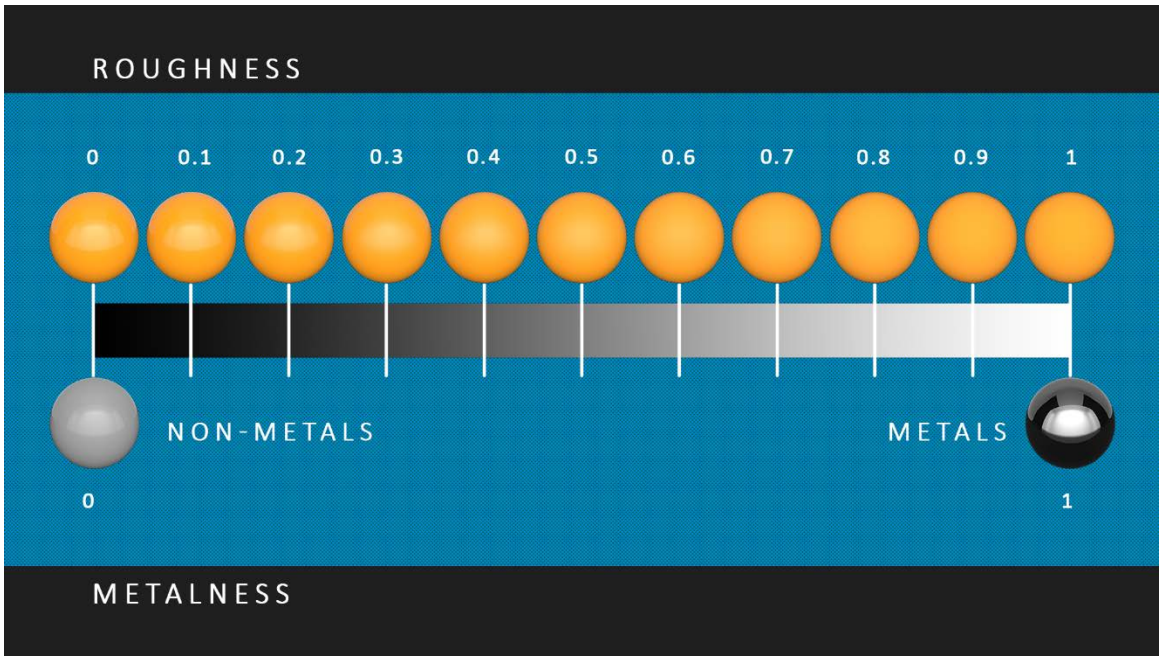

Figure 9. Roughness (up) and Metalness (down) difference in values (Lappa 2017)

In comparison to Specular & Glossiness workflow, Metalness & Roughness has lesser control over the reflectivity of the conductors, but it makes more accessible for the artists to simulate the smoothness. As both Maps are in grayscale, they use less texture memory, making it better for the optimization.

3.2.3 Capturing Reflection

There are a number of ways of projecting the reflection of the scene in video games. The most common methods are based on real-time and static capturing, where the results have their advantages and disadvantages.

One of the methods is called Cube Mapping. Based on 3D Game Textures (Ahearn 2008, 115), Cube Maps are based on 6 six seamless images arranged into a cube geometry, covering all axis of the reflection, which result is demonstrated by Epic Games in Figure 10. The texture is always static and gives approximate parallax for the observation, but does not reflect any dynamics of the scene, and thus should ever be placed correctly.

With the increased capabilities of the physical rendering, modern game engines, like Unreal Engine 4 (Epic Games, n.d. d) provide a real-time reflection capturing via Reflection Capture Actors which can be seen in Figure 10. The position of the actors redraws the images of the Cube Map every frame dynamically. With this method, players can see the movable object being reflected.

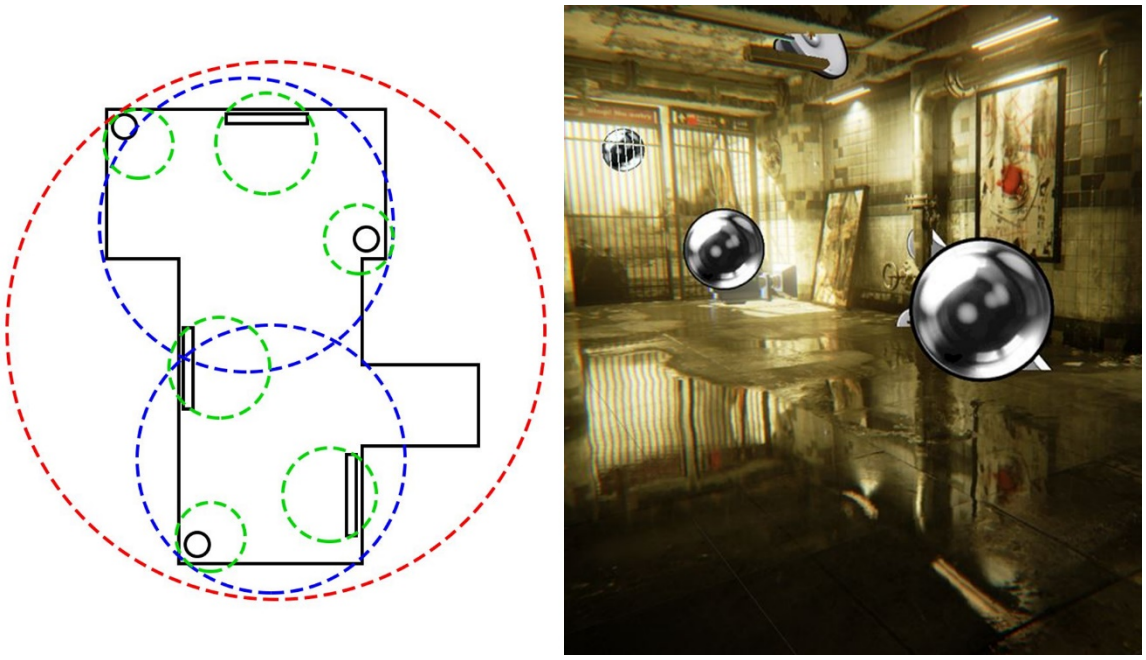

Figure 10. Reflection Capture scheme (left) and scene (right) (Epic Games, n.d.)

With the constant recording of the environment, the real-time capturing process requires a tremendous amount of the computer resources and should always be placed on the scene cleverly. As Figure 10 illustrates, the actors are put in different ways. Some of them capture the whole environment, while only a few are placed to achieve the mirroring with more density. Combined with the static reflection via the Cube Maps, both processes can project the whole scene accordingly to the player position, where the prime spots are reflective,

leaving the rest for the pre-baked information and better optimization (Ahearn 2008, 115).

3.3 Normal & Displacement Maps

Normal Mapping in 3D is a technique that is used to fake the lighting of bumps and dents instead of adding more polygons to an object. According to Ahearn (2008, 117), the texture is used to make a model appear like high polygon model with various details with smoothed edges to define a relief of the simple mesh for the lighting and look more realistic and appealing.

In real-time rendering, the relief of the surface that is stored in the Normal Map is in tangent space. At that point, according to Polycount (2017), the light that comes in contact with the surface of the texture image is analyzed from its position and, due to this action, Shader decides which color pixel and with what intensity will be displayed.

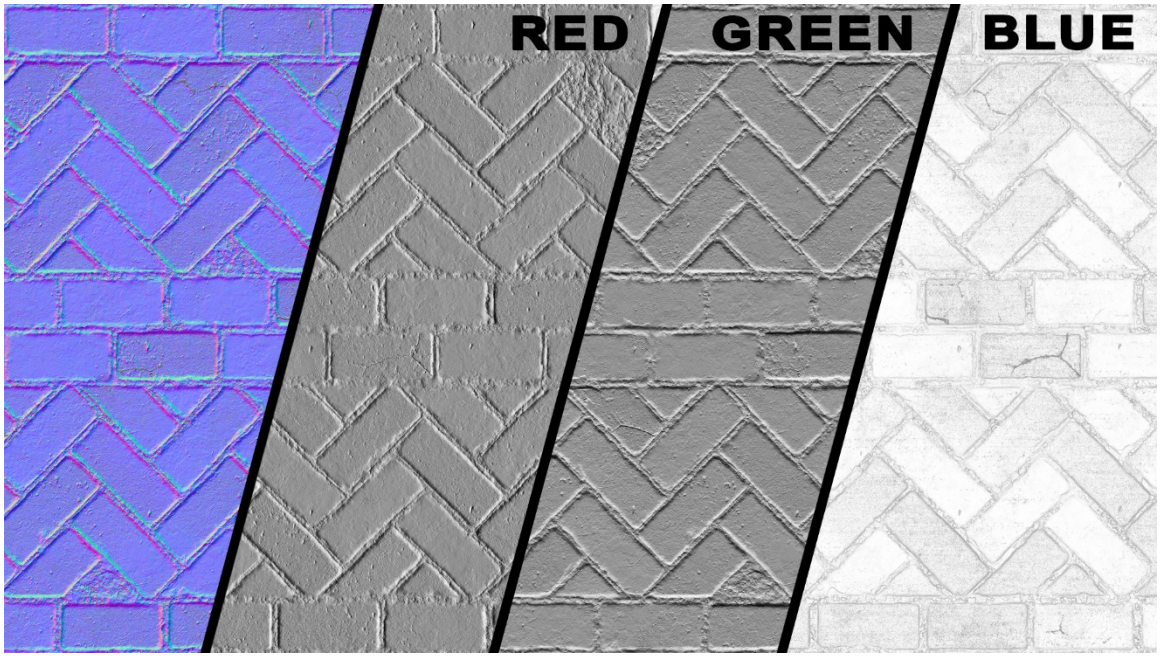

Figure 11. Normal Map example with Red, Green and Blue channels exposed (Lappa 2017)

As Figure 11 demonstrates, Normal Maps are constructed from 3 channels: Red, Green, and Blue, where each color corresponds to the three-dimensional coordinates. The red channel in the image indicates the direction of light along

the X axis (right and left), while the green channel is responsible for the Y axis (up and down) and the blue channel keeps the Z axis which directs the position of normals outwards and inwards of the surface in Normal Map.

Height Map, also called Displacement Map, has a similar idea of normal Mapping. However, it is more complicated and can be very slow to render and should be used within reason.

With the introduction of a tessellation method, Displacement Map became a subject of the interest among many artists. According to Nvidia (n.d.), the process uses a Height Map to define the displacement of the vertices by dividing polygons into many pieces as can be seen in Figure 12.

Figure 12. Height Map vertex displacement with tesselation applied (Lappa 2017)

Figure 12 also shows that the texture is in greyscale and is used in combination with Normal Maps to give an extra definition to surfaces where the surfaces are responsible for rendering large bumps and displacements. The brighter pixels of the image make higher elevations while, the darker pixels do the opposite, where middle grey pixels make no change.

3.4 Light Maps

With the ability to produce light in real-time rendering, game engines provide a variety of options how the shadows of environments are stored in a scene.

While Diffuse or Base Color Maps are mostly for the color information only, Light Maps are there to support the missing data in the packed textures. The textures are based on the similar workflow like the environment Map, where it contains non-updatable information of the light values and is multiplied with the base textures for static conditions. (Polycount 2015b.)

The baked lighting information provides a variety of the possibilities to keep the shadows and light bounce information with a low amount GPU resources, but since static lights only use Light Maps, they use an increased amount of texture memory. Their shadows should be re-rendered before any changes are made inside the scene, which means that they cannot record moving objects in real-time. (Unreal Engine 2017.)

The second method that is available in modern game engines is a possibility of dynamic lighting rendering, where the shadows and lights are changed upon the movement and kept inside of the environment without any light textures recorded. With the update of every frame, this method requires more significant GPU performance for the satisfactory results. (Unreal Engine 2017.)

3.4.1 Ambient Occlusion

An Ambient Occlusion Map (AO) is used to create exposed soft shadows. It is a grayscale texture that keeps the shadowing information of finely accessible places such as slits, corners, and cracks. (Epic Games, n.d. a.)

The process of making an Ambient Occlusion Map is tied to storing the information from an individual geometry in a static condition. According to Epic Games (n.d. a), it can either be stored in a greyscale image, or it can be rendered inside of the game engine in real-time. Based on Polycount (2015a), this method

of an in-game shadowing is called Screen Space Ambient Occlusion. The process dynamically captures the required details, but it requires a more decent graphics card to proceed flawlessly and can involve some issues like shading errors.

Figure 13 illustrates more information of how Ambient Occlusion texture is stored. The base color on the left represents the color information, while all small shadows are separated on a different surface with white value covering most of the space.

Figure 13. Base Color Map (left) with Ambient Occlusion Map (right) extracted (Lappa 2017)

3.4.2 Emissive

Another texture to cover is called Emissive Map. It is used to simulate a self-illumination of the surface by increasing the intensity of the specific pixels. The Map builds the appearance of being lit up which is frequently used as screens, light bulbs, and more. (Unity 2017b.)

Emissive Map receives areas, where black cover most of the space except the parts that need to glow with an intensity of the values and colors in RGB color space. The example of this method can be seen in Figure 14, where XAMK logo

was chosen by the author to build it as a mask with color transitioning to black on the edges. As a result, the area of a color in texture is glowing but does not illuminate the light even at the higher intensity.

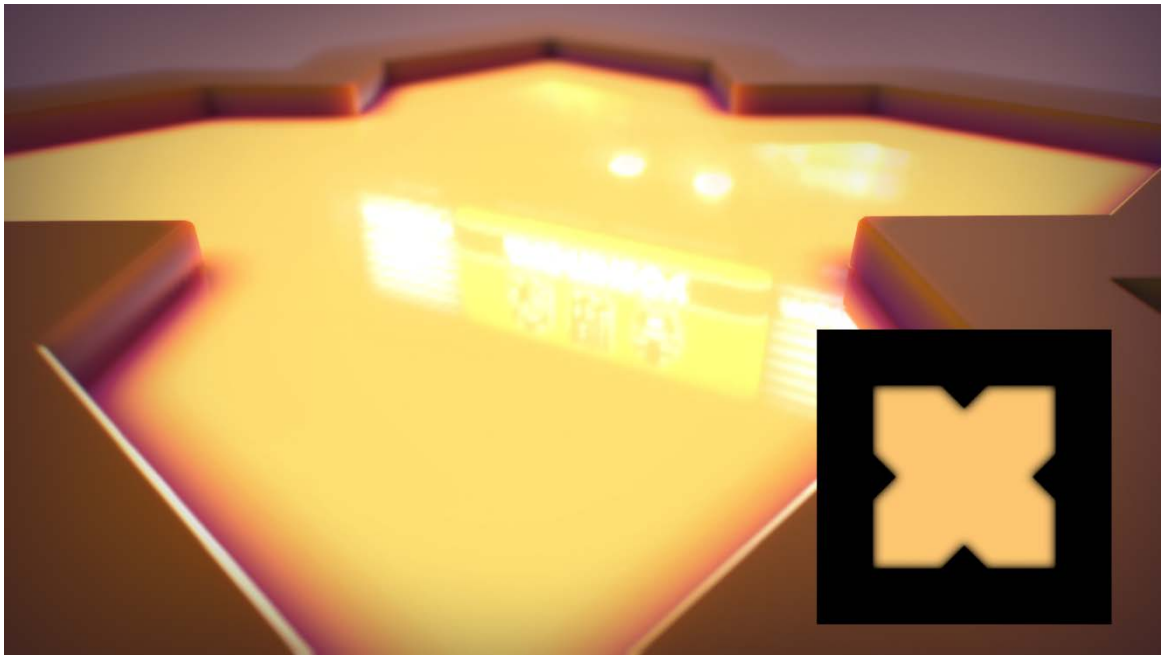

Figure 14. Example of Emissive Map applied (Lappa 2017)

3.5 Transparency Map

In game engines, when creating a material with an ability to see through it, a transparency Map is there to control the rendering of a passage of light. It is also referred to Opacity Map or Alpha Map and is usually set in an extra alpha channel of Base Color in RGB format. (Unity 2017a.)

For the demonstration purpose, Figure 15 demonstrates a render of a box with a gradient Map applied to show the parameters of the transparency control. Black (0) is responsible for the full Transparency, while white (1.0) considers the fully-visible area. The transitions of the values in the middle of the gradient show the varying opacity in between two constant values, done in greyscale mode.

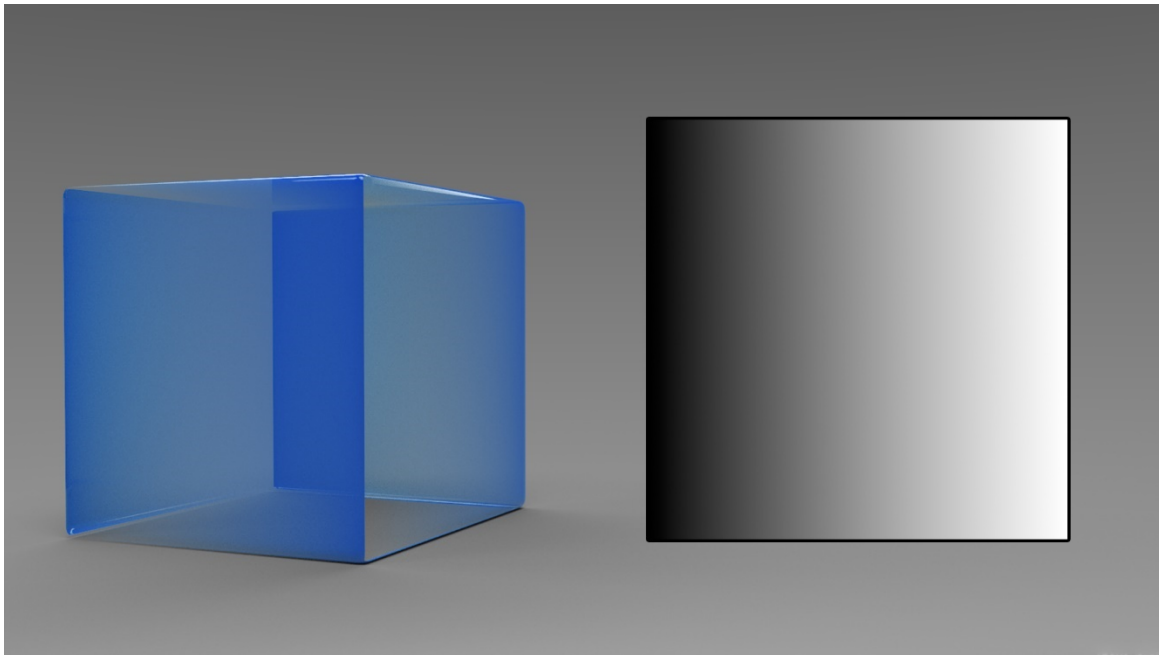

Figure 15. Example of the usage of an 8bit linear gradient as Opacity Map (Lappa 2017)

When talking about the convention of the values between whites and black, it is necessary to take into a consideration of bit depth of the images. According to Wallstrom (2015), bit depth is the number of value data storage used to represent in color channels.

The value data delivers the difference in the smoothness of the opacity transitions. A 1-bit texture is only black and white. It has its use for the lowest memory usage but comes with a significant loss of the quality. A 4-bit image in comparison has a slighter better variety of the transition, which allows blending in 16 colors. The gradation between the values, as seen in Figure 15, allows to break the edginess and uses 8-bit transpose. Combined with the texture filtering, it provides to get a full range of greys with 256 colors, which makes it useless for building the Maps with higher color values for even more smoothness of the edges. (Polycount 2015d.)

3.6 Detail Map

When texturing an object, some details at the closer look may become broken and pixelated, resulting in a low quality of the textures. Detail Map was introduced to hide the squarish visuals and add more quality to the surface. It builds an

illusion of the higher resolution of the image and can be seen in Figure 16 in comparison to a stated issue.

Figure 16. Difference between simple texture (left) and Detail Map applied (right) (Lappa 2017)

Made by the author, Figure 16 also demonstrates the way how the textures are used. By adding a tileable noise across the surface, Detail Map builds a visually higher resolution. The effect is achieved by overlaying the detail texture on top of the base material with an extended UV channel.

Based on Polycount (2015c), a Detail Map is a method that is used to define details in close view. The technique is built in the Shader uses a higher amount of UV tiling, which makes smaller pixels to appeal to the viewer. Adjusted, real-time rendering allows the texture to fade out at a medium distance to hide tiling artefacts which can significantly save a texture memory in keeping the primary materials in lower resolution.

4 APPROACHING PHOTOREALISTIC MATERIALS

A Material is an asset in a game engine that can be used to define the visual look of a mesh in the scene by an artist (Epic Games, n.d. b). Knowing how to make the right parameters, it defines the type of surface from which the object appears

to be made of to the eyes of the players by the help of physically-based rendering.

The process of building an appealing texture for its further realization across the whole game environments relies on a variety of applications. While it is essential to follow the high standards of the industry, artists must be able to apply the production of the textures for the needs of the development.

This chapter is divided into several sub-sections that focus on texture building methods with multiple software combinations. The overall goal was to bring game-ready materials, which were created by the author to demonstrate the workflows.

4.1 Sculpting

Sculpting in both game and movie productions became important over many years. The method is used to produce a high poly model that is difficult to provide via traditional 3D modelling and is used by the professionals to achieve both realistic and organic results.

There are many applications available for the industry which allow achieving digital sculpts including 3D coat, Autodesk Mudbox, and Pixologic Zbrush, which is the 3D industry’s standard nowadays and grants the possibility to build the object organically. For the demonstration purpose, the author sculpted a rock material, that has much complexity in the natural forms that can easily appeal to the viewer.

Pixologic Zbrush is a software that allows using customizable brushes to shape a virtual clay in a modern way. For the textures, it comes very handy to sculpt high polygonal meshes and then translate the information into a two-dimensional image for a further production.

Before the start, it is essential to consider the size of an object. Since Zbrush is an application that is used to reproduce an object in 3D space, the process

should always rely on a scale reference to keep the proper resolution of the required result. For this reason, a simple plane has been created in Autodesk Maya to fulfil that purpose. The object serves merely to guide an artist for both size and tile of the sculpt, that can remarkably simplify the post-production after the capturing the details.

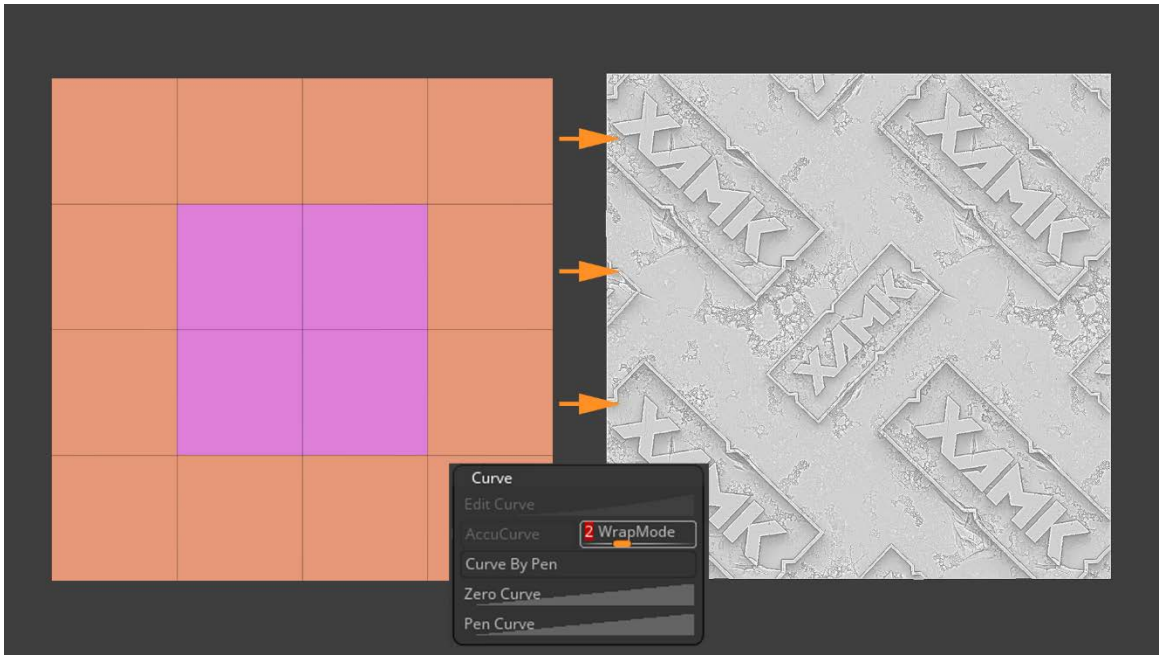

Figure 17. Reference Plane (left) with a tiled sculpt (right) (Lappa 2017)

The organic workflow is a relatively simple and a straightforward process. Mostly it is based on building the low polygonal objects and then further reinforcing it and tweaking them inside ZBrush with a large sub-division level for a high detailing. Following these steps, the mesh can be created in a short time with the result as it seen in Figure 17, where a logo of XAMK was used as a demonstration of a custom-made brush.

The rock base (Figure 18) was created by the combination of simple cubes and duplicating them in the space of a plane. With the satisfied position, to make it tile it requires filling the sides, which can be achieved by using deformation – offset tool. With a positive or negative value of 100 in a required axis of the offset, adding tileable details inside of the border can be quickly set up by increasing the number of warp mode by 2 (Figure 17).

The detailing process heavily relies on the mixture of the brushes and sub-division within Zbrush, which is more creative, rather than technical. Since the final resolution of the texture will be 2048x2048, there is no reason to achieve the hyper-realism of the shapes, where the smallest dents will become pixelated.

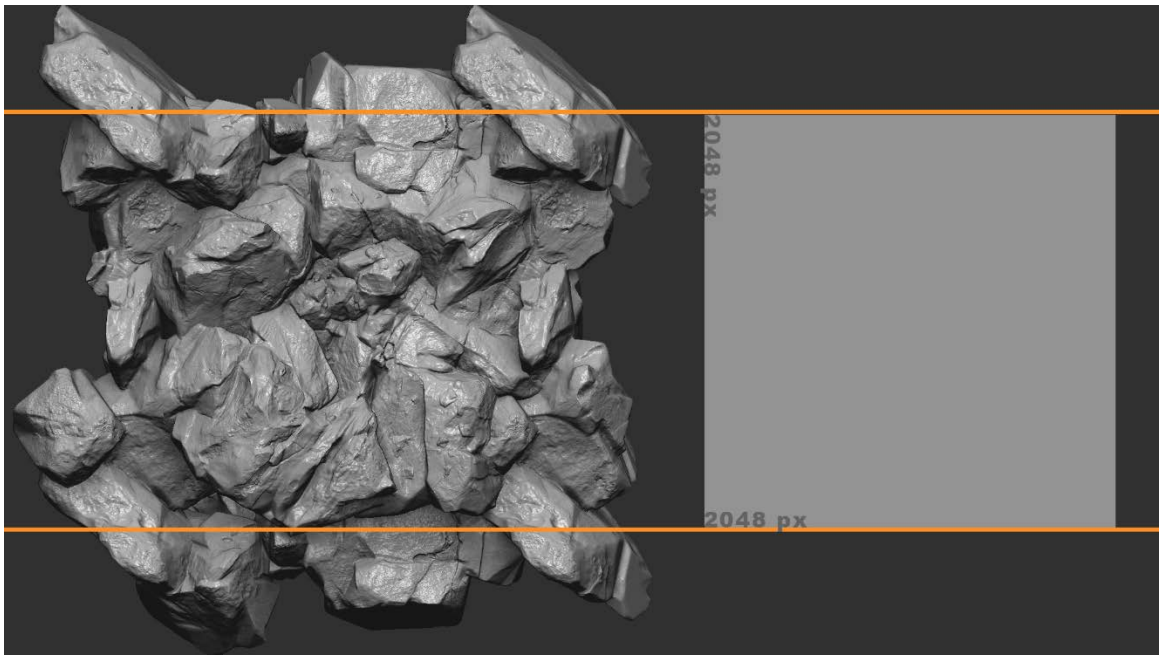

Figure 18. A final tiled rock sculpt with a scale reference (Lappa 2017)

When the work is finalized (Figure 18), the information of the surfaces can be captured in many ways. The simplest and the fastest one is to render the image straight from the canvas with a Normal material applied. The render must be set-up with the highest quality of the distributed mesh with the shadows turned off. When the calculation is over, the image can be easily saved from documents and export window.

The overall process is simple but requires more post-production of the images, including correction of Normal Map and fixing tiling issues if they are. The organic shapes can be created extremely fast, but for a better result, the pipeline can demand much more computer power.

4.2 Image-based Texturing

Approaching photorealism requires an enormous effort in producing the right appealing details. One of the ways of creating it is straight from a photo image and fit it for the correct usage in Physically-based Rendering.

Since Physically-Based Textures are bitmaps, they can be easily created by any programme that allows editing images. One of the world's best software that is used among the many artists – Photoshop, developed by Adobe. It supports to adjust and create the photographs, graphic designs, artwork, and with the extended plugins like Quixel Tools can become a full application with a wide variety of 3D texture editing. While knowing the parameters of each Texture Map, Photoshop is a perfect option for tweaking them by hand.

When creating a texture, it is essential to start with gathering references. There are plenty resources of the images available for artists on the internet including google.com or textures.com. The most prominent con of making a texture via Photoshop is that artist can reproduce a material straight from the source. Figure 19 illustrates how the reference can be manipulated for building a unique look with the shadows being extracted.

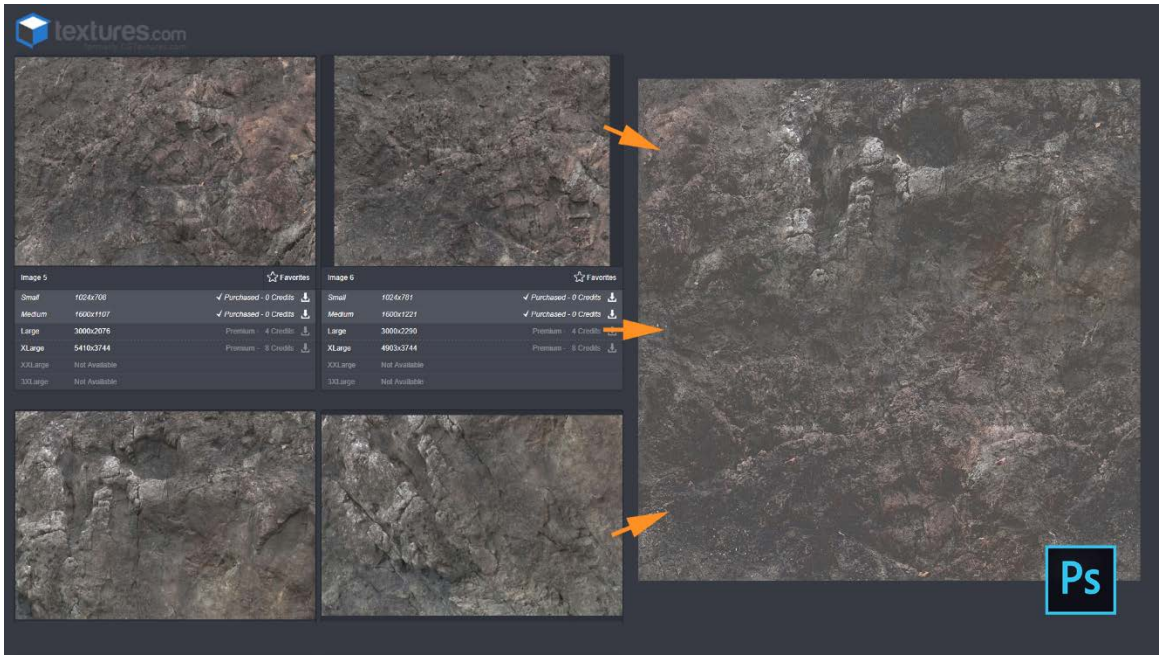

Figure 19. Reference images built into a Base Color Map in Adobe Photoshop (Lappa 2017)

As a result, Figure 20 illustrates the combination of the results of 3D sculpting and Photoshop texturing applied on top with all the physical parameters. Notwithstanding, the pipeline allows to build the materials in no time, but due to the rasterization of the images, the process is limited to its size resolution.

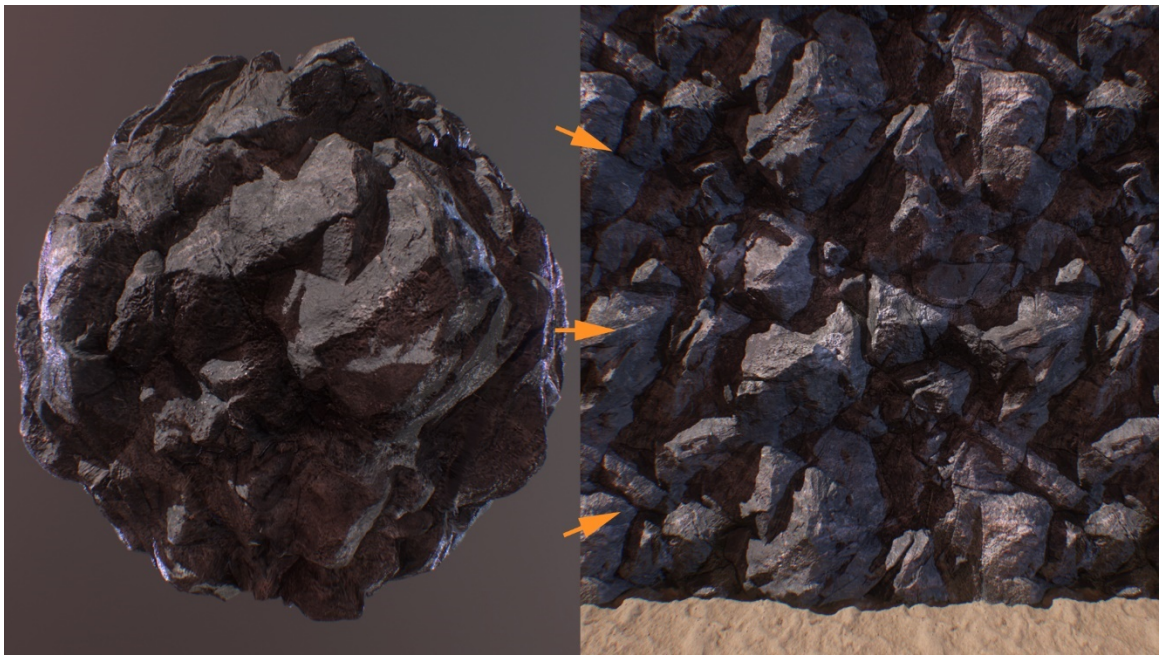

Figure 20. Textures done in Adobe Photoshop applied to a mesh (Lappa 2017)

4.3 Procedural Texturing

Procedural texturing is another way to approach the building of a texture. While raster graphics are fixed at a resolution limit, the bitmaps created procedurally have no restrictions.

Substance Tools is a set of application consisting of Substance Designer and Substance Painter. Developed by Allegorithmic, the technology allows building procedurally generated textures with the required set of defined bitmaps.

4.3.1 Substance Designer Workflow

Substance Designer is a node based tool with a procedural approach to materials. With this workflow, artists have many possibilities of texture production without the loss of data, where the changes, applied to any of nodes, will

automatically be applied to an end-result. It allows using mathematical functions to build dynamic texture that has no limitation in size resolution with being tiled.

Substance Designer pipeline consists of a primary setup with blocking out the basic shapes, with further incorporation of the details. As the process is finalized, the visualization can be distributed as the physically-based textures in the nodes called outputs.

As can be seen in Figure 21, the workflow requires much more effort and further node optimization. Creating a realistic material can be time-consuming, and the process can become complicated to achieve a proper result from the reference.

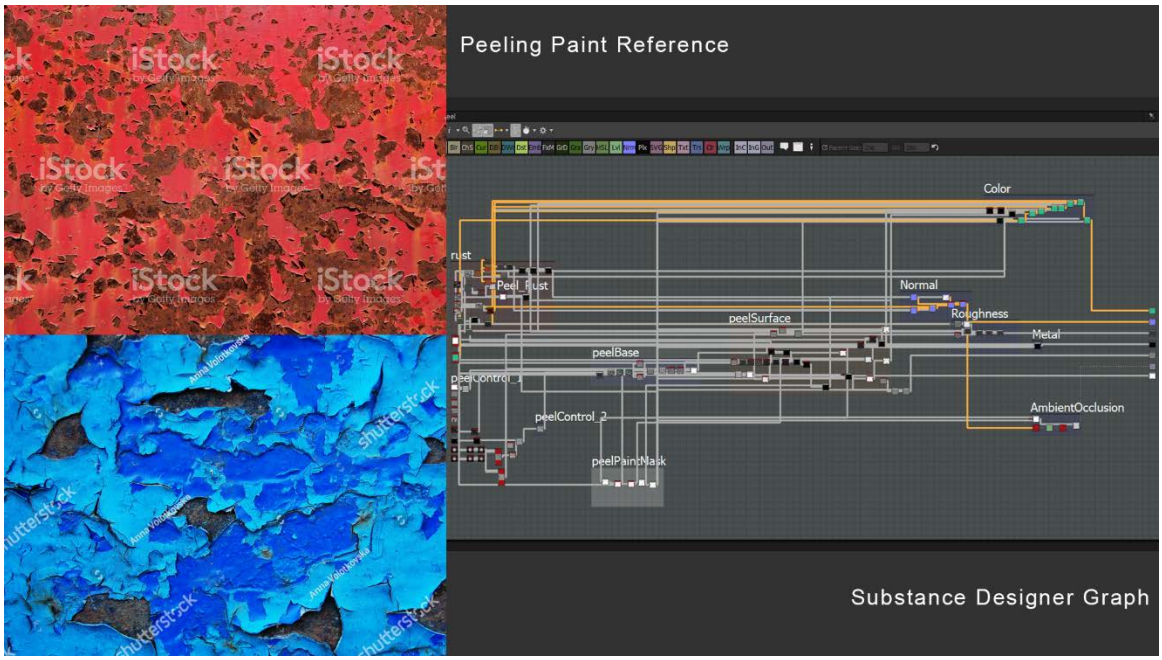

Figure 21. References (left) and Substance Designer graph (right) of the material (Lappa 2017)

On the other hand, since the process is entirely procedural, the variety of the visual properties can be adjusted in many ways. Many impressive visual effects are possible which is only limited by the artist's imagination. These features can be anything from simple patterns to fully textured surfaces.

As an example, Figure 22 demonstrates the possibilities of how flexible material adjustment can be. With the extended node functions, Substance Designer can

allow an artist to tweak the parameters, including attaching a bitmap as a mask, which dynamically can fit into a material.

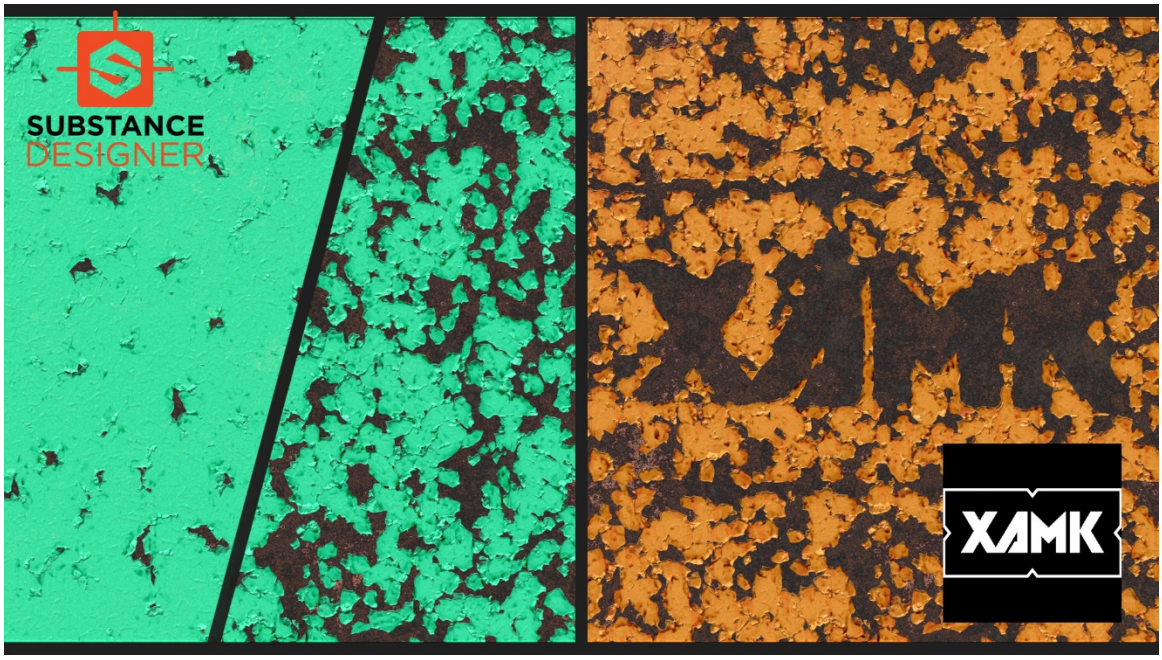

Figure 22. Substance Designer material exposed parameters (Lappa 2017)

4.3.2 Substance Painter Workflow

Sometimes it can take a considerable effort to create certain types of assets with specific features in a procedural workflow. Substance Painter is a handy software that allows to place, paint or generate details in certain places with multiple options, where artists can arrange on their own in any creative way.

The application is like Photoshop when it comes to organization. Taking the material from substance designer allows adjusting the workflow dynamically and in a short time. Based on the author's texture production (Figure 23), the process can be divided into three steps – baking, masking, and tweaking.

The baking process allows creating a position coordinates of the mesh with several Maps, such as Normal, Ambient Occlusion, and more. Based on these textures, Substance Painter allows building adjustable masks to create dust in small dents, rust of top or peels on the edges. For the unique information, the software also provides painting on top of the mesh with different parameters, not

depending on any data of the baked textures. A demonstration of the whole process can be seen in Figure 23.

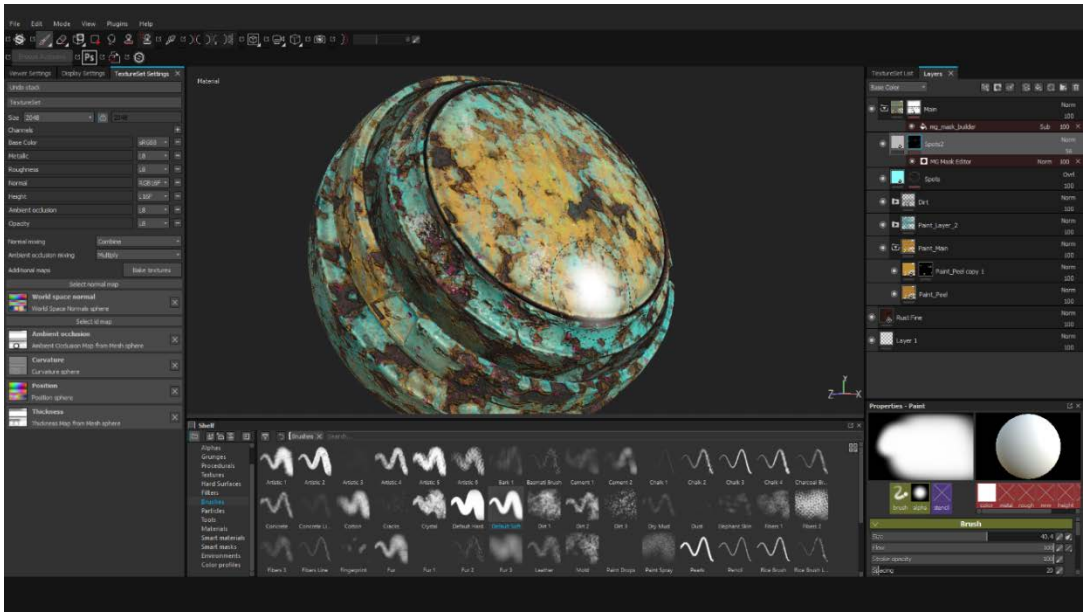

Figure 23. Texturing in Substance Painter (Lappa 2017)

With the functions developed in Substance Designer, a base material creation brings a comprehensive variety to Substance Painter. Figure 24 demonstrates both results, where the process was done in Substance Applications by the author, where the pipeline used the same material with the different parameters of the exposed functions.

Figure 24. Substance Designer material (left) and textured material in Substance Painter (right) (Lappa 2017)

4.4 Photogrammetry

Photogrammetry is a method of capturing multiple overlapping photographs and taking measurements from them to create 3D models of objects or scenes. The technique brings a possibility to build a highly accurate and realistically photo-textured models of structures, landscapes, and objects.

Photogrammetry is gaining in popularity and usage since it produces impressive results comparable to laser 3D scanning technologies at much lower costs. A few resources were picked to achieve the required results, including a full-frame photo camera Canon EOS 6D to take images, a Canon 50mm f/1.8 prime lens that can produce the pictures with the lowest distortion and a software called Agisoft Photoscan for building a 3D mesh.

As it can be viewed in Figure 25, the process is considered in picturing multiple photographs like a panorama and combining them in a three-dimensional space. The software estimates the position of a camera to build a high poly mesh in X, Y, and Z coordinates.

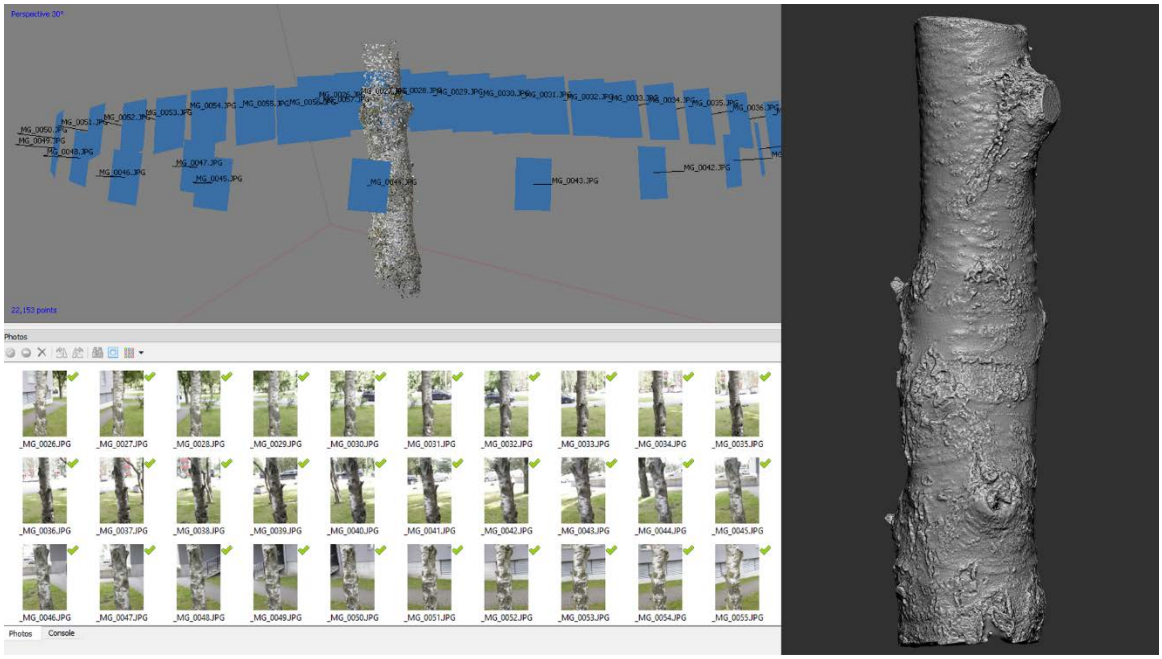

Figure 25. Photo scanning (left) and a high-poly result (right) (Lappa 2017)

Though Photogrammetry allows building a mesh with a texture information, the process alone is not enough for the further realization in real-time rendering. The next stages are required to transfer the detail to the textures, which could be realized in multiple ways, including sculpting for the topology corrections, baking the Color and Normal Maps and fixing the results either in Photoshop or Substance Tools. With all the fixes applied, the results can be seen in Figure 26.

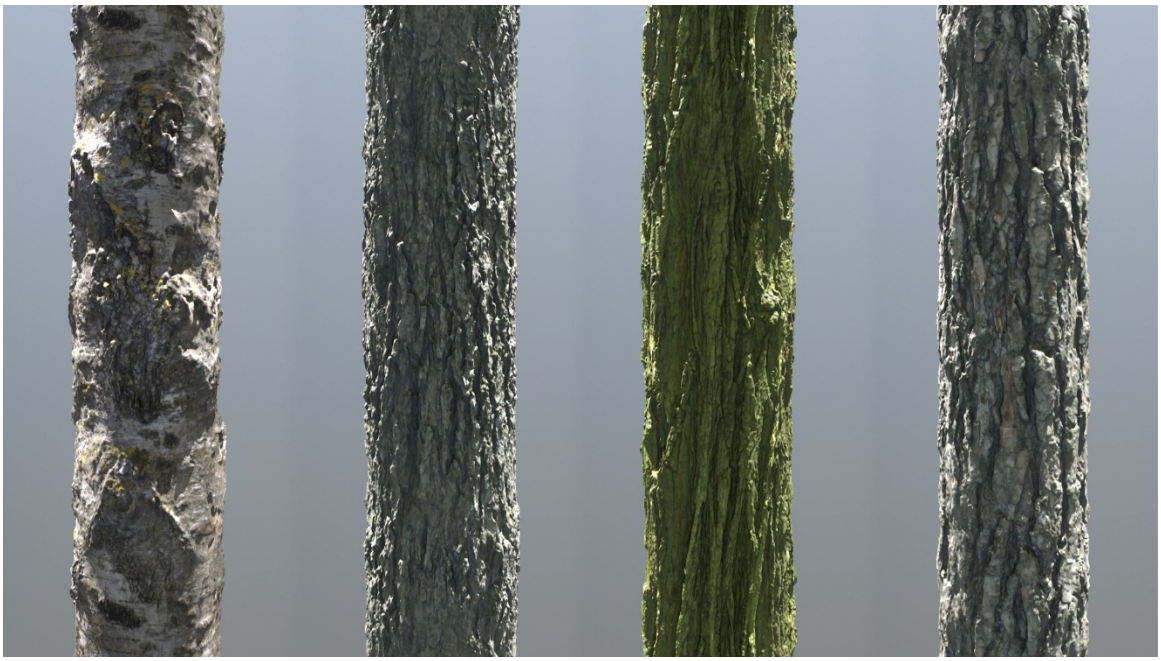

Figure 26. Finalized photogrammetry tree barks (Lappa 2017)

With the possibility of scanning the objects, the production of the unique and realistic textures becomes more available than ever. Though the process requires the knowledge of the mixed usage of various software, taking the images straight from life can drastically decrease the time of building the appealing material.

5 CONCLUSION

The development of textures dates back to the last century. For decades, 3D has evolved, until it became a standard for the gaming industry. The construction required many optimizations, in which the texturing succeeded and grew to the stunning results in the simulation of realism in real-time rendering.

Based on the sources and personal experience, the author was able to demonstrate some frequently used textures in various examples that were created specifically for this thesis. Also, the author showed not only how each property of surfaces carries a role, but despite the appearance of varieties in the parameters, the process does not require in-depth knowledge of mathematics to achieve the desired outcome.

Further, after the study, the author was able to demonstrate various ways of creating realistic textures. All renderers were successfully done in the shortest time and were presented in the thesis as examples of using not only individual programs but also their combinations.

In conclusion, despite the fact that the textures and programs have reached the necessary level to transform realism in real-time, most of the work depends on the artist himself. Every year the number of tools is growing, but only a creative approach can create an unprecedented level of work in the shortest possible time.

REFERENCES

Adobe. No Date. Color Theory for the Desktop. The Nature of Light and Color. [online] Available at: http://dba.med.sc.edu/price/irf/Adobe_tg/color/light.html [Accessed: 10 October 2017].

Ahearn, L. 2008. 3D Game Textures: create professional game art using Photoshop. 2nd edition. Burlington: Elsevier, Inc.

Blinn, F. J. & Newell, E. M. 1976. Texture and Reflection in Computer Generated Images, Graphics and Image Processing, Vol.19, No.10, October 1976. Available at: https://pdfs.semanticscholar.org/0196/090590638e3063410563386a917c27dae975.pdf [Accessed: 8 September 2017].

Chapman, G. 2015. Utah Inventions: The birth of computer graphics, August 19, 2015. Available at: https://www.ksl.com/?nid=1012&sid=36039333 [Accessed: 8 September 2017].

Epic Games. No Date a. Ambient Occlusion. [online] Available at: https://docs.unrealengine.com/latest/INT/Engine/Rendering/LightingAndShadows/AmbientOcclusion/ [Accessed 25 October 2017].

Epic Games. No Date b. Materials. [online] Available at: https://docs.unrealengine.com/latest/INT/Engine/Rendering/Materials/index.html [Accessed 25 October 2017].

Epic Games. No Date c. Physically Based Materials. [online] https://docs.unrealengine.com/latest/INT/Engine/Rendering/Materials/PhysicallyBased/ [Accessed 25 October 2017].

Epic Games. No Date d. Reflection Environment. [online] Available at: https://docs.unrealengine.com/latest/INT/Resources/Showcases/Reflections/ [Accessed 25 October 2017].

Jaquays, P. & Hook, B. 1999. Quake III Arena Shader Manual Revision #12. [online] Available at: http://toolz.nexuizninjaz.com/Shader/ [Accessed: 8 September 2017].

Jones, A. Z. 2017. Cathode Ray History, August 10, 2017. Available at: https://www.thoughtco.com/cathode-ray-2698965 [Accessed: 8 September 2017].

Kusher, D. 2003. Masters of Doom: How Two Guys Created an Empire and Transformed Pop Culture. 1st edition. New York: Random House, Inc.

McDermott, W. No Date a. The Comprehensive PBR Guide by Allegorithmic - vol. 1. Light and Matter : The theory of Physically-Based Rendering and Shading. Available at:

https://www.allegorithmic.com/system/files/software/download/build/PBR_Guide_Vol.1.pdf [Accessed: 8 September 2017].

McDermott, W. No Date b. The Comprehensive PBR Guide by Allegorithmic - vol. 2. Light and Matter: Practical guidelines for creating PBR textures. Available at: https://www.allegorithmic.com/system/files/software/download/build/PBR_volume_02_rev05.pdf [Accessed: 8 September 2017].

McDonald, E. 2017. The global games market will reach $108.9 billion in 2017 with mobile taking 42%, April 20, 2017. Available at: https://newzoo.com/insights/articles/the-global-games-market-will-reach-108-9-billion-in-2017-with-mobile-taking-42/ [Accessed: 8 September 2017].

Nvidia. No Date. DirectX 11 Tessellation. [online] Available at: http://www.nvidia.com/object/tessellation.html. [Accessed: 9 October 2017].

Pantone. No Date. How Do We See Color?An introduction to color and the human eye. Available at: https://www.pantone.com/how-do-we-see-color [Accessed: 9 October 2017].

Pernady, F. 1996. Ray-Casting Tutorial For Game Development And Other Purposes. [online] Available at: http://permadi.com/1996/05/ray-casting-tutorial-table-of-contents/ [Accessed: 8 September 2017].

Pettit, N. 2015. The Beginner’s Guide to Physically Based Rendering in Unity. 17 November 2015. Available at: http://blog.teamtreehouse.com/beginners-guide-physically-based-rendering-unity [Accessed: 25 October 2017].

Policarpo, F. & Oliveira, M. M. & Comba, J. 2005. Real-Time Relief Mapping on Arbitrary Polygonal Surfaces. ACM SIGGRAPH 2005 Symposium on Interactive 3D Graphics and Games, Washington, DC, April 3-6, 2005, pp. 155-162. Available at: http://www.inf.ufrgs.br/~oliveira/pubs_files/Policarpo_Oliveira_Comba_RTRM_I3D_2005.pdf [Accessed: 8 September 2017].

Polycount. 2017. Normal Map Technical Details. [online] Available at: http://wiki.polycount.com/wiki/Normal_Map_Technical_Details#Tangent-Space_vs._Object-Space [Accessed: 9 October 2017].

Polycount. 2015a. Ambient occlusion map. [online] Available at: http://wiki.polycount.com/wiki/Ambient_occlusion_map [Accessed: 9 October 2017].

Polycount. 2015b. Light map. [online] Available at: http://wiki.polycount.com/wiki/Light_map [Accessed: 9 October 2017].

Polycount. 2015c. Texture types. [online] Available at: http://wiki.polycount.com/wiki/Texture_types [Accessed: 9 October 2017].

Polycount. 2015d. Transparency map. [online] Available at: http://wiki.polycount.com/wiki/Transparency_map [Accessed: 9 October 2017].

Russel, J. 2015. Basic Theory of Physically-Based Rendering. [online] Available at: https://www.marmoset.co/posts/basic-theory-of-physically-based-rendering/ [Accessed: 8 September 2017].

Shahrani, S. 2006. Educational Feature: A History and Analysis of Level Design in 3D Computer Games, April 25, 2006. Available at: https://www.gamasutra.com/view/feature/131083/educational_feature_a_history_and_.php [Accessed: 8 September 2017].

Shklyar, D. 2004. 3D Rendering History. [online] Available at: http://www.cgsociety.org/CGSFeatures/CGSFeatureSpecial/custom_story/1647&page= [Accessed: 24 September 2017].

Unity. 2017a. Albedo Color and Transparency. [online] Available at: https://docs.unity3D.com/Manual/StandardShaderMaterialParameterAlbedoColor.html [Accessed: 10 October 2017].

Unity. 2017b. Emission. [online] Available at: https://docs.unity3D.com/Manual/StandardShaderMaterialParameterEmission.html [Accessed: 10 October 2017].

Unity. 2017c. The Fresnel Effect [online] Available at: https://docs.unity3D.com/Manual/StandardShaderFresnel.html [Accessed: 10 October 2017].

Unreal Engine. 2017. Lighting with Unreal Engine Masterclass | Unreal Dev Day Montreal 2017 | Unreal Engine. [video online]. 9 October 2017. Available from: https://www.youtube.com/watch?v=ihg4uirMcec [Accessed: 10 October 2017].

Wallstrom, C. 2015. 8-bit vs 16-bit – What Color Depth You Should Use And Why It Matters. [online] Available at: https://www.diyphotography.net/8-bit-vs-16-bit-color-depth-use-matters/ [Accessed: 9 October 2017].

Whitted, T. No Date. An Improved Illumintaion Model for Shaded Display. Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.107.3997&rep=rep1&type=pdf [Accessed: 8 September 2017].

Williams, A. 2017. History of Digital Games. 1st edition. Boca Raton, FL: Taylor & Francis.

Wolf, J. P. M. 2008. The Video Game Explosion: A History from PONG to PlayStation and Beyond. 1st edition. London: Greenwood Press, Inc.

Wilson, J. 2015. PBR Texture Conversion Referenced. [online] Available at: http://www.marmoset.co/toolbag/learn/pbr-conversion [Accessed: 8 September 2017].

LIST OF FIGURES