Bachelor's thesis

Information and Communications Technology

2020

Jaani Nordberg

VISUAL EFFECTS FOR MOBILE GAMES

– creating a clean visual effect for small screens

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

BACHELOR'S | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information and Communications Technology

2020 | 38 pages

Jaani Nordberg

VISUAL EFFECTS FOR MOBILE GAMES

- creating a clean visual effect for small screens

The importance of visual effects (VFX) in video games is significant. They are meant to support the gameplay and bring the in-game world alive. They can be explosions, leaves falling from trees, or flashing shapes in the user interface that tells the player what to do. The job of a VFX artist is to create all of these. VFX artists need a broad understanding of visual effects in general, but also the skills to use a variety of software and techniques to create them.

This thesis aimed to make a clean and clear visual effect for a mobile game developed by Flatfish Games Oy but also improve the skills needed in creating visual effects and understanding different creation techniques. The purpose of the effect was to visualize the ultimate ability of the game character and at the same time create a basis for how different character's own character and elemental variations of the same ultimate ability would work and look.

The thesis provided a comprehensive picture of what skills and tools are needed to make visual effects and what to consider when making visual effects for mobile games. All the processes of creating the effect, from design to implementation, were reviewed and explanations of why the chosen solutions had been arrived at were given. The effect was done with the Unity game engine utilizing Unity's built-in particle system, Blender 3D modeling software, and GIMP image manipulation software. The effect made was clear and supported the gameplay. It was also hoped that the thesis would be useful for other beginners or people who want to work with visual effects.

KEYWORDS:

visual effect, VFX, Unity, particle system, video game graphics

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tieto- ja viestintätekniikka

2020 | 38 sivua

Jaani Nordberg

VISUAALISET TEHOSTEET MOBIILIPELILLE

- puhtaan visuaalisen tehosteen luominen pienille näytöille

Visuaalisten tehosteiden (VFX) merkitys videopeleissä on merkittävä. Niiden tarkoitus on tukea pelin pelattavuutta ja elävöittää pelin sisäistä maailmaa. Ne voivat olla räjähdyksiä, puista putoavia lehtiä tai vilkkuvia muotoja käyttöliittymässä, jotka kertovat mitä pelaajan kuuluu tehdä. VFX-artistin tehtävä on luoda kaikki nämä. VFX-artistit tarvitsevat laajaa ymmärrystä visuaalisista tehosteista yleisesti, mutta myös taitoja käyttää erilaisia ohjelmistoja ja tekniikoita niiden luomiseksi.

Tämän opinnäytetyön tavoitteena oli tehdä puhdas ja selkeä visuaalinen tehoste Flatfish Games Oy:n kehittämään mobiilipeliin ja samalla parantaa osaamista visuaalisten tehosteiden luomisessa ja erilaisten luomistekniikoiden ymmärtämisessä. Tehosteen tarkoitus oli visualisoida mobiilipelin hahmon erikoisabiliteettia ja samalla luoda pohja miten eri hahmojen omat hahmo- ja elementaalivariaatiot samasta erikoisabiliteetista toimisivat ja näyttäisivät.

Opinnäytetyö antoi kattavan kuvan mitä taitoja ja työkaluja visuaalisten tehosteiden tekemiseen tarvitaan ja mitä ottaa huomioon tehtäessä visuaalisia tehosteita mobiilipeleihin. Mobiilipeliin tehdyn efektin kaikki prosessit suunnittelusta toteutukseen käytiin läpi ja kerrottiin, miksi valittuihin ratkaisuihin oli päädytty. Efekti tehtiin Unity-pelimoottorilla hyödyntäen Unityn sisäänrakennettua partikkelisysteemiä, Blender 3D-mallinnusohjelmaa ja GIMP-kuvan manipulointiohjelmaa. Tehty tehoste on selkä ja tukee pelin pelattavuutta. Toivomuksena oli myös, että opinnäytetyöstä olisi hyötyä muille aloitteleville tai visuaalisten tehosteiden pariin haluaville ihmisille.

ASIASANAT:

visuaalinen tehoste, VFX, Unity, partikkelisysteemi, peligrafiikka

CONTENTS

PICTURES

LIST OF ABBREVIATIONS (OR) SYMBOLS

3D mesh	A Collection of vertices, edges and faces that create 3-dimensional shape.
Addon	Separtate component that adds a new feature to an existing software.
Diffuse	Light reflecting technique in computer graphics.
Gradient	Smooth transition of a color.
Rigging	Creating a skeleton to a 3D or a 2D object to create an animation.
Rendering	Process of creating a 2D image of a 3D mesh or a scene.
Sprite	2D game object.
Texture	An image used to give color information to the surface of a 3D mesh.
UI	User interface.
VFX	Visual effect.

1 INTRODUCTION

Due to game engines like Unreal Engine and Unity creating games has never been easier and the amount of learning material found free from the internet means that almost anyone can start making games. Because of this, the importance of standing out for consumers and publishers has become even more important. Graphics are usually the first thing possible customers see and the graphics alone can be the reason behind the decision to use their money and time for the game. Visual effects are there to boost the graphics and gameplay and give life to the game world. Therefore, visual effects play an extremely important role in today's game industry.

This thesis will go through different techniques on how to create visual effects and what principles there are for creating them. It'll also go through the industry-standard tools along with independent developer-friendly tools. The purpose of this thesis is to tell what goes into making visual effects and show how a visual effect for an arcade sports mobile game was created. My goal is also to give myself and hopefully others some insight on what to learn and become a better VFX artist.

The visual effect was created for a mobile game made by Flatfish Games Oy and they act as both my employer and the commissioner of this thesis. The visual effect needed clearly to visualize the action, support the gameplay avoid cluttering the small mobile screen. The effect also needed to act as a base for future similar effects with variations of different elemental and character-specific aspects like ice and slime.

Before joining Flatfish Games in 2019 I had no experience with visual effects and by summer 2020 I had worked on various small visual effects for different projects. It's important to understand that learning how to create visual effects is a very long process due to various techniques and software and there doesn't exist a correct way to create them. The result is what matters. It should convey the action to the player whether it's rain falling from the sky or an explosion. The techniques I learned came mostly from trial and error and analyzing different games and online tutorials.

2 VISUAL EFFECTS IN GAMES

The founder of Beyond-FX and a VFX artist Keith Guerrette describes that the visual effects artist's job is to add motion to everything that are not characters or vehicles. Water drops falling on the window on a rainy day or a campfire lighting a dark forest. Visual effects are essential in storytelling. They bring life to the world and immerse the player into it. Visual effects can guide the player or give vital information. The role of a VFX artist is getting more and more important. (Guerrette 2016)

The game industry has only been utilizing visual effects from the early 2000s as the hardware used had the capability to handle particles in much bigger amounts than earlier. For example, Valves Source game engine introduced particle editor in 2007. Before that visual effects were often hard-coded and used sparsely. Movies on the other hand have been utilizing visual effects for a much longer time. There are VFX companies that employ hundreds of people and within these companies, there are often different departments only focusing on a certain part of the visual effect. One department could only be focusing on rigging and one in 3D modeling. In the game industry, visual effects are either outsourced or created with a small team of technical artists and VFX artists. This means that the individuals need to have much wider skills on the tools and techniques used to make visual effects but also knowledge and deep understanding in general. (Guerrette 2016) (Valve Developer Community 2020)

2.1 Practical and Symbolic Effects

Visual effects in games can be split into three categories; Practical effects, Symbolic effects, and something between them. Effects like dust, rain, or fog are considered highly practical effects as they give the game world movement and life. They also help immerse the player into the character's world and ground the character into its world. Practical effects can be considered as narrative effects as they often give some insight into a character or the surrounding world. Green-colored guts and gore can tell that a character is an alien or some sort of monster, and explosions and blasts give you information about what is happening around the player. They are meant to be seen by the in-game character as much as the player. (Grissom 2018)

Practical effects can also be used to guide the player through levels. We are hardwired to avoid things like flames, lava, acid, and other harmful things so it's an effective way to show the player where not to go.

Picture 1. Flames used to guide the player. (theRadBrad 2017)

Symbolic effects are meant to guide the player, give information about what happened and what to next. They are meant to be seen by the player and not by the in-game character. They are often UI elements like a simple number popping to the screen after an attack was performed telling the player how much it dealt damage or a button that is highlighted with a flashing light reminding the player to push it. (Grissom 2018)

Picture 2. Numbers used to inform the player how much an attack dealt damage. (OMEGZKI GAMING STYLE 2020)

Between these two types of effects are a variety of effects that can be considered to belong to both of these types. They are both giving information to the player about the mechanics of the game but are still rooted in the reality of the character. These types of effects are often seen in competitive multiplayer games where it's important to understand the character's abilities and how far they extent but also giving insight about the character. Overwatch and League of Legends are good examples where a lot of the visual effects fall between the practical and symbolic. For example, in Overwatch characters like Winston and Reinhardt have a shield they use to protect the team. They are both categorized as tanks so the shield represents the character as it's protecting other players but also gives information about the shape and size and how much it can take damage. (Grissom 2018)

2.2 Artistic Principles of Visual Effects

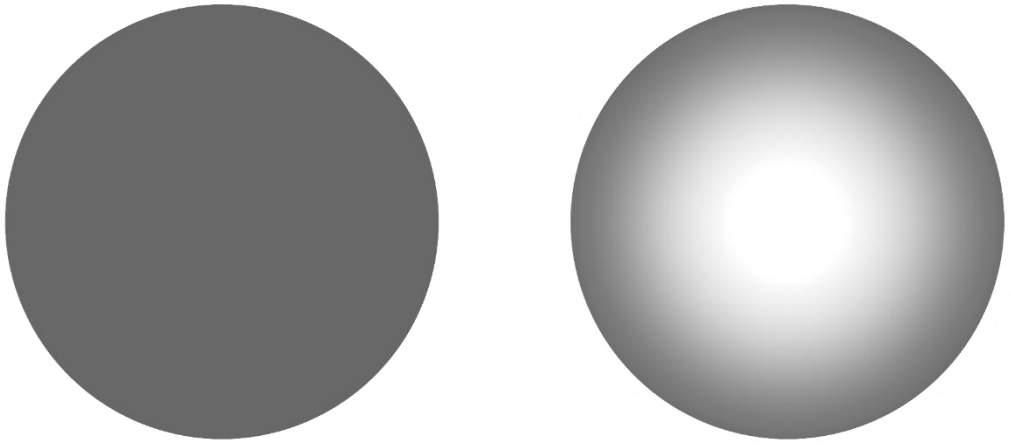
2.2.1 Shape

If a player sees a sharp-looking thing like a spike the player understands that it's something that should be avoided. Shapes give a lot of information and the shapes that we are used to seeing in our daily lives are often the same shapes that are used to give information to players in games. A red cross is a symbol that is used by hospitals and it's also very often used by health packs in games to heal damage. An arrow is a simple way to give information about where to go. Shapes are very universal but it's important to remember that different shapes can have different meanings in different cultures. Shapes also give information about the area of effect. Like in Overwatch the character Soldier: 76 has the ability to heal other players with a device that casts an area to the ground where players are healed. The healing area is visualized by a simple circle. Inside the circle players get healing and outside it, they don't. (Chamberlain & Keyser 2017)

Picture 3. Shape of the healing area in Overwatch.

2.2.2 Contrast

To help the shapes give even more information to the player it's good to make them stand out. By contrasting certain parts or effects it's easier to draw the player's focus to avoid danger or guide to safety. A bright white arrow in the middle of a dark forest can't be missed and the player should understand where to go but of course, there are more subtle ways to do this. Gradients are also a good way to show for example how much damage an attack makes from a distance. If an attack deals more damage the closer the player, the contrast is higher from the center and fades out further. (Chamberlain & Keyser 2017)



Picture 4. Example of using gradients for contrast.

2.2.3 Color

There are many ways how colors and color values are used to communicate and inform the player. A high color value means brightness and is often interpreted as high heat. For example, a flame is usually bright yellow or even white in the center and darker orange on the outer parts. Changing the colors of a flame from yellow and orange to be more greenish means that there might be something different in that flame. Maybe something magical. The shape and movement of an effect play a big role in how the

color is interpreted. A light blue orb moving slowly doesn't look dangerous but a violently moving light blue thunder looks. Red blood is associated with damage, but a red cross is associated with healing which is opposite of each other. (Chamberlain & Keyser 2017)

2.2.4 Timing

Timing can be hard to explain in words. Timing can be simply thought as to when an effect is executed or what happens to the effect over time. The effect could change color or size over time. For example, variations of speed can make a big difference in how an effect feels. If you make a projectile move from point A to point B with a linear velocity it feels and looks slower compared to making the projectile first move a little slower and make it rapidly increase the velocity even when the time used to get from A to B is the same. This can be very effective in stylized and cartoonish games but doesn't work in realistic-looking games. A bullet should have a linear velocity unless it feels odd. Anticipation is also a good way to make the effect feel different. In realistic games a grenade just explodes but if you add some movement before the explosion like stretching or shaking the anticipation grows. The timing is highly tied to the gameplay. Randomizing the different scales and lifetimes of the elements inside of the visual effect can also give a more natural feeling. (Chamberlain & Keyser 2017)

3 TOOLS AND TECHNIQUES USED FOR CREATING VISUAL EFFECTS

In games, visual effects are often created with the game engine's built-in particle systems but particle systems usually need custom-made textures, texture sheets, or meshes. These need to be created with third-party software like Autodesk 3Ds Max, Adobe After Effects, and Photoshop among other similar software. There are various ways and various tools that can achieve the same result with very different techniques. For example, water splashes can be created with fluid simulations or texture sheet animations. There isn't a correct way to approach a visual effect because the result is what matters. If the effect supports the gameplay the technique of how the result was achieved isn't that important. Even though the result is what matters an artist should always consider what is the easiest and most optimized approach to achieve the effect. Much like in coding there are many ways to achieve what is needed but the easier and more optimized way is usually better. They should be treated like puzzles.

3.1 Common Visual Effects Techniques

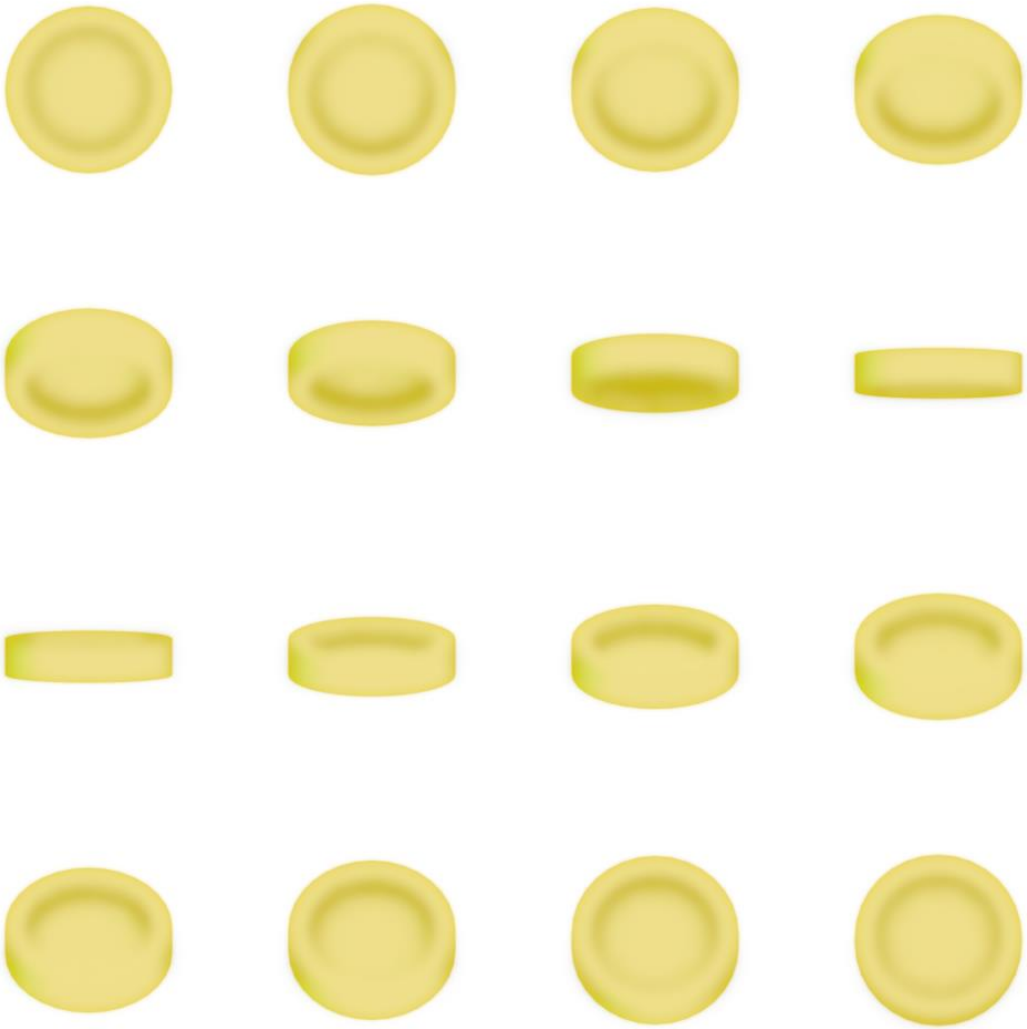
3.1.1 Sprites

An easy way to create simple visual effects is to use sprites and add some movement to them. For example, if you spawn multiple images of smoke at a small area and add some upward movement and randomness with a particle system or animating it can look very convincingly like rising smoke clouds. Or a plasma gun could use a single image of a dot that is slightly blurred it looks like a glowing projectile. This is a simple and fast way to create either placeholder effects for testing or even for the final product.

3.1.2 Texture Sheet Animations

One of the most common types of visual effects are texture sheet animations. They are animations that are rendered or drawn into a single 2D image. For example, you can have a 2048x2048 pixel image that has sixteen 512-pixel images in it. The images are played frame by frame in order. There are many ways to create them. They can be hand-drawn in a photo manipulation program or rendered in 3D software among many other techniques. Texture Sheet Animations have the same principle that was and still is used to make cartoons. Everything from realistic-looking flames to stylized explosions can be created with this technique. This technique can be extremely light compared to how good it can look but great texture sheets are often difficult to make and takes a long time to master. (Thorn 2015)

Here's an example of a sprite sheet I made for an effect that is triggered when a loot box is opened. It's a coin flip animation rendered in 3D software and edited in photo manipulation software to add color and a more cartoony look.



Picture 5. Texture Sheet Animations of a Coin Flip.

3.1.3 3D Meshes

3D meshes are useful when a visual effect requires a specific shape or simply if the effects need more depth and its viewed from different angles. For example, a character can use a sword that is caught on fire. Texture Sheet Animations are also often used as a texture in 3D meshes. A waterfall is a good example. In a 2D game, the waterfall is seen only from one angle, so it doesn't need any shapes but in a 3D game it usually needs a mesh that is shaped like a waterfall and a texture sheet to visualize the water's

movement on it. Another good way is to create a shader instead of the texture sheet that visualizes the movement of the water on the mesh.

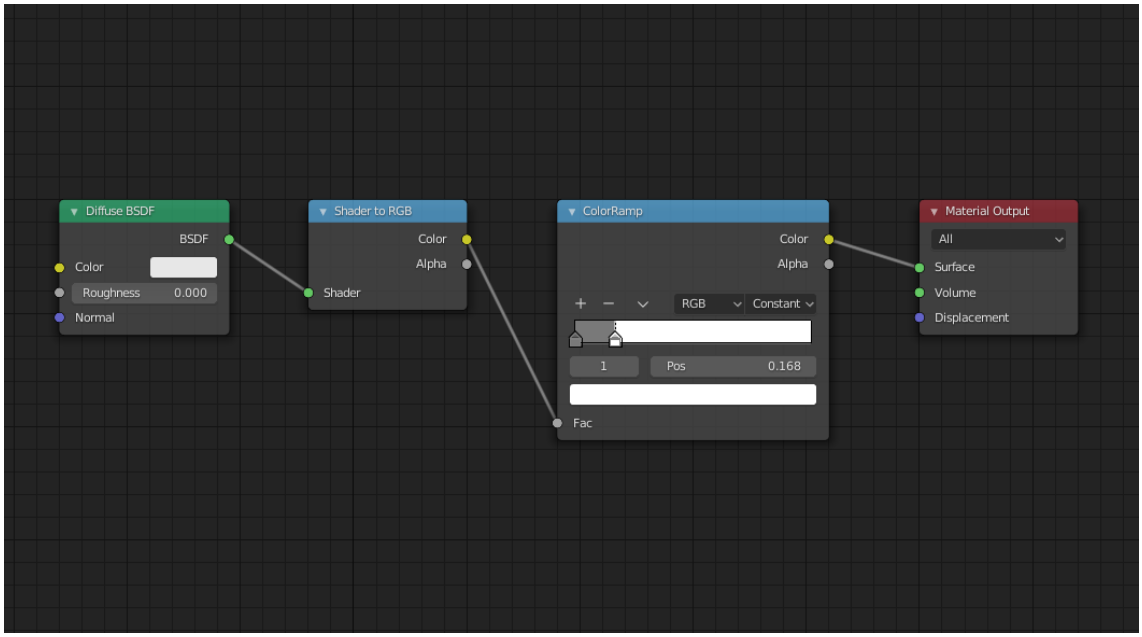
3.1.4 Simulations and Volumetric Lighting

Real-time simulations and volumetric lighting are the future of visual effects but as of now, they are mostly used in offline rendering software like 3D modeling software due to being very demanding on hardware. Simulations like liquids are constantly getting better optimized and newer hardware is better and better at handling them in real-time. It is very possible that in the very near future effects like blood splattering and water splashes are all real-time rendered. The same is most likely going to happen with volumetrics. Volumetric lighting is used to simulate how light behaves when it hits things like dust or fog. Game engines like Unreal Engine 4 already has the capability to create volumetric clouds. (Oberbeck 2017)

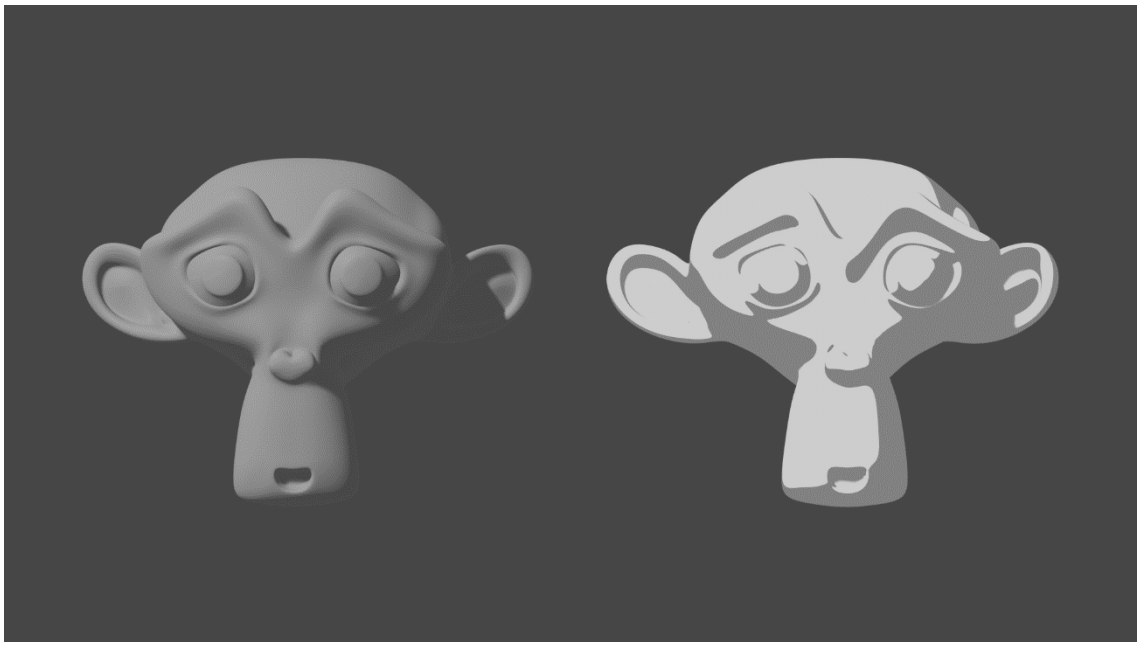
3.2 Shaders

In basic, a level shader is a set of instructions that tell how the Graphics Processing Unit (GPU) renders a pixel. For example, cel-shading is a shading technique where all shadows of a 3D model are shaded in a way that shadows color value above a certain value is always rendered in one single color and values below the certain value in one with no color and then multiplied to the base color or color texture of the material. This gives the 3D model a very cartoony look because the shadows are linear and single color without fading in any part of the model. Shaders can also distort the geometry of a mesh. Water shaders often have the shader to create the look of the water but also create the movement of the waves.

Here's an example of a diffuse shader I modified into a simple cel-shader in Blender using Blenders node-based shader scripting and the result rendered compared to a typical diffuse shader.



Picture 6. Modified diffuse shader in Blender to create a cel-shader.



Picture 7. Comparison of diffuse shaded and cel-shaded 3D model.

Dissolve shader is one of the most used shader types for visual effects. There are a lot of different kinds of looking and differently made dissolve shaders but they all do the same thing. As the name suggests they dissolve a mesh or a sprite to a different material or completely away. It's often used when an object or character is destroyed, burned, or teleported but the possibilities are endless. The way it works is that a color value between one and zero where white is one and black is zero determines what pixels are discarded. An animated black and white texture like a noise texture is used to control the values. (Alisavakis 2017)

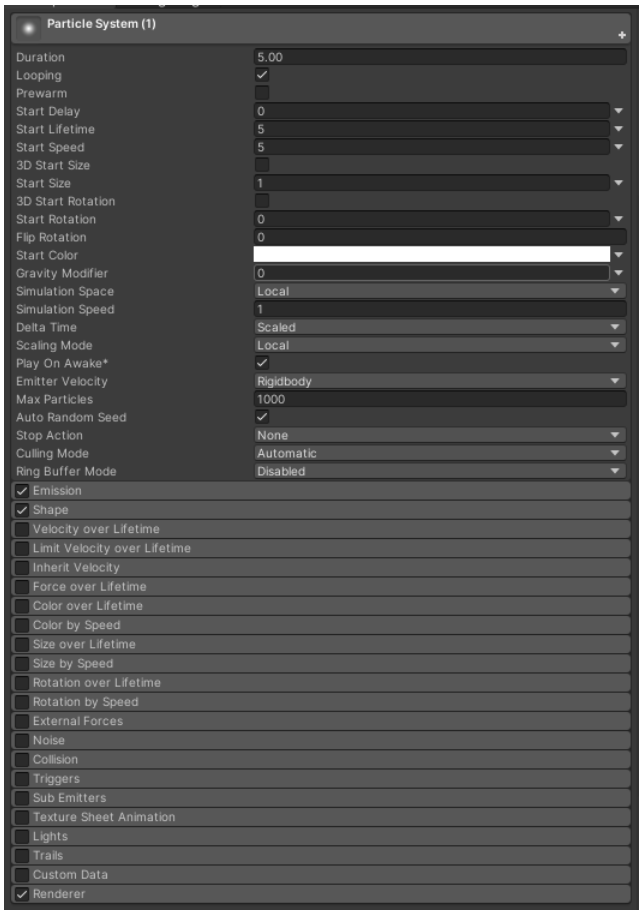
Picture 8. Example of a dissolve shader. (GentleGaming 2019)

Shaders can be extremely useful in visual effects and the movie industry has been using shaders to help in creating visual effects for much longer than the game industry. Star Wars Episode IV: A New Hope used shaders in 1977. Shaders are usually written using Microsoft developed HLSL (High-Level Shading Language), NVIDIA developed Cg (C for Graphics), or GLSL (OpenGL Shading Language) that was developed by the OpenGL Architecture Review Board (ARB). These are all C-like languages. Shader coding can be hard to learn but it's an extremely useful skill to have when making visual effects. Some tools like Blender and Unity have the possibility to write the shader code using

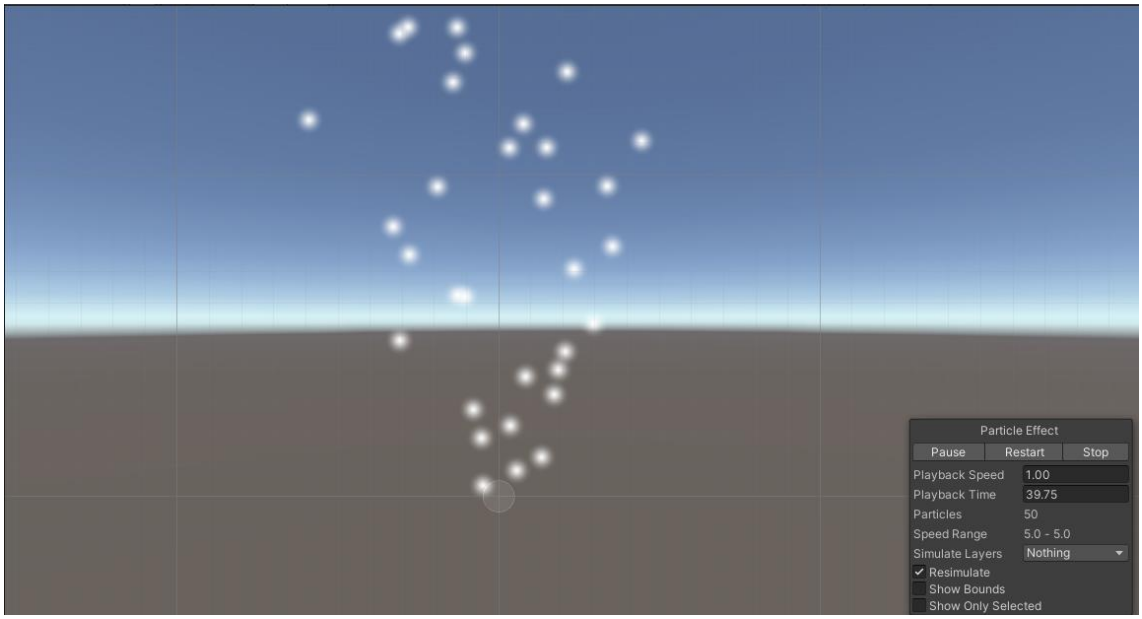
node-based visual scripting that can be easier for beginners. (Bailey & Cunningham, 2009)

3.3 Particle systems

A particle system is a tool that is used to simulate single particles are either 2D images or 3D meshes. A particle system can be thought of as a command center controlled by the user that gives the single particle or a collection of particles a set of rules on what to do. For example, when to spawn or how fast it moves during its lifetime. The easiest way to explain a particle system is to use Unity's particle system as an example because most of the particle systems work the same but might have different modules inside them to control the individual particles. Also, the effect described later was created with Unity's particle system so to understand the effect the particle system needs to be explained in detail.



Picture 9. Particle system in Unity and all the modules used to control the particles.



Picture 10. Default particle system in Unity scene.

Most of the values within the particle system can be set to a certain integer or a random value between two integers giving randomness to the effect. The values can also be set with a curve or a random value between two curves giving different values to the particle during its lifetime.

Main module is the main control panel that defines the duration of how long the whole particle system runs, the size and rotation of the particles, strength of the gravity, and other basic settings. (Unity 2020)

Emission controls the number of particles emitted. It can be set to a constant stream of particles or for example to a burst of various amounts of particles. (Unity 2020)

Shape defines the shape of the volume where particles are emitted from. Turning off this module means that all of the particles are emitted from a single point to a single direction. If the shape module is turned on the particles are emitted from an area that is defined either from prebuilt shapes or custom shapes from meshes. For example, a sphere

shape emits particles in every possible direction and a plane to only one direction but within a larger area than the single point. (Unity 2020)

Velocity over Lifetime and **Limit Velocity over Lifetime** controls the velocity of the particle. For example, it can be set to accelerate the particle or to dampen the velocity on a certain axis over the lifetime of the particle. (Unity 2020)

Color over Lifetime and **Color by Speed** controls the color and transparency of the particle over its lifetime or by the velocity it is moving. For example, the particle could start as white and in the middle of its lifetime turn to black and fade away completely before dying. (Unity 2020)

Size over Lifetime and **Size by Speed** are basically the same as Color over Lifetime and Color by Speed but controls the size of the particle. (Unity 2020)

Rotation over Lifetime and **Rotation by Speed** are the same as Size over Lifetime and Size by Speed but controls the rotation of the particle. (Unity 2020)

Noise creates a turbulence effect to the particles. It can be set to high values to create chaos or smaller values to give lifelike movement to leaves falling from trees. (Unity 2020)

Collision. The particles can be set to collide with everything. The collider is limited to a certain shape that makes it tricky to use in some situations. (Unity 2020)

Sub Emitters are basically particle systems within the particle system that trigger for example when the original particle hits a target or simply dies. (Unity 2020)

Texture Sheet Animation controls the speed and amount of frames that are played from a texture sheet. (Unity 2020)

Trails add a trail to the particle like fireworks or a spaceship that has a flame coming out of its rocket. (Unity 2020)

Renderer controls how the individual particles are rendered. You can decide if you use meshes as particles or billboards how are they aligned and sorted to the screen. The material and shading options are also decided here. With the renderer module turned off nothing is visible on the screen. (Unity 2020)

3.4 Common Visual Effect Tools

3.4.1 Adobe After Effects and Photoshop

Adobe has a variety of industry-standard software that are used from photographing to web design. Visual effects are one of them. After Effects is a great tool to create 2D texture sheets and UI effects because of its powerful particle system and other various tools. To give a more stylized and unique feeling to texture sheets or create sprites to be used in particle systems they are often hand-drawn. Photoshop is a widely used software for this. For a VFX artist, the skill to use Photoshop is often a requirement. (McDonald 2020)

After effects and Photoshop can be expensive tools for independent game developers and hobbyists. As for After effects, there doesn't really exist a free alternative tool but for Photoshop an open-source photo manipulation software GIMP is a great alternative. It has a lot of the same features that Photoshop provides.

3.4.2 Autodesk Maya and 3Ds Max

These are both 3D modeling software and they are both made by Autodesk. Even though they are very similar they have some differences. Maya has better tools for animation and rigging and 3Ds Max has better tools for modeling. These are both the industry-standard tools that are used in most of game development companies. But like most of the industry-standard tools, these are very expensive, and they have a big learning curve. (McDonald 2020)

Blender 3D is an open-source 3D modeling software that is raising its head to compete against Autodesk's products. It has a lot of useful tools for creating visual effects in addition to the modeling tools like animating, simulations, and a particle system. The release of version 2.8 has made even bigger game development companies contemplate switching to Blender.

4 WHAT TO CONSIDER WHEN MAKING A VISUAL EFFECT FOR MOBILE

Visual effects are equally important regardless of the platform on which the game was made for. Whether it's a mobile game or a console game visual effects can in some situations even make or break the game. There are a lot of things to consider when making visual effects for games. The mobile game market is dominated by casual and hyper-casual games that are often highly stylized or simplified in their art style. Candy Crush is a good example of a casual game where the player instantly understands how the game works and gets into playing quickly. This is why they are called casual games as they don't require the player to spend long periods of time learning the mechanics. Casual games like Candy Crush are extremely popular because of marketing and making the games highly addictive. Visual effects play also a large role in making the gameplay as addictive as it is. Candy Crush is very good at satisfying and rewarding the player's actions with visual feedback.

4.1 Optimization

There is a reason for a lot of games being stylized as it's much more forgiving than a realistic art style. If a game has realistic human models and the animation aren't exactly how humans would move it easily comes out as looking uncanny. Stylized art doesn't have to follow real-world rules thus giving the artist much more freedom. Stylizes effects and graphics are also more forgiving in the number of vertices that the meshes need and the resolution that the textures and sprites need to have. This affects a lot of the build sizes and performance. Optimization is very important in mobile games because mobile devices are also not built equal. The goal should always be to make as good-looking effects as possible with minimum impact on performance.

As great of a technique texture sheet animations are, they can have a negative effect if used too extensively especially on mobile devices. The image sizes need to have a big enough resolution for the animation to be clear. Texture sheets also often need to have transparency in some parts. For example, in the coin flip texture sheet shown earlier all the parts where there is no color are transparent. All the pixels that are behind the transparent parts of the texture are shaded twice even though the transparent parts are

not visible at all. This can have a heavy impact on performance. Using shaders is also a great way to create visual effects but it should be considered that the more complicated the shader is the more it has an impact on the performance. There are also shaders that work on some mobile phones but doesn't work on others due to differences in hardware. (Unity 2020)

4.2 Screen Size and Other

Mobile screens are smaller than monitors and television screens and this has to be considered when making visual effects. It's easy to make the mistake of using a lot of time making a very detailed and complicated effect just to come to the conclusion that most of the details are not even clearly visible or the effect is taking too much space on the screen. This is often the case if the effects are not tested regularly on a mobile phone and only tested inside the game engine that is running on a much bigger screen. Having a small screen has its advantages too like the lower resolution images and sprites don't look as they are bad in quality compared to a bigger screen where you see the details much more clearly. (Encz 2015)

The lighting is good to take into consideration. Mobile games usually use precalculated lighting and shadows for optimization meaning that objects don't receive or cast any shadows in real-time. There are of course some exceptions where some characters or moving objects do cast and receive shadows, but these can often be faked in different ways. Especially in 3D mobile games if the effect is a 3D model it can look out of place when all the surrounding world has shadows, but the effect doesn't.

It's also important to have a good plan. All of the above should be considered as early as possible to avoid unnecessary use of time. Understanding the limitations in the planning phase can save hours of work. The same goes for testing. The effect should be tested on the devices as early and as often as possible. Getting feedback is also important. Artists tend to become blind to their creations and might spend lots of time on effects that visually look good but don't support the gameplay.

5 CASE: VISUAL EFFECT FOR A ARCADE SPORTS MOBILE GAME

5.1 Planning

The effect was to visualize the character's ultimate ability where the character shoots a fiery ball to the opposite player's side. When the ultimate ability was activated there was a short time for the player to aim the shot and the character would play a looping animation. I wanted that the effect would make the player feel strong as much as to make the character look powerful. The fiery ball already had a good looking and feeling effect, so I decided to only focus on the effect to make the character look powerful. As this was an effect that would have similar variations to different characters, I wanted to find a common base idea that all characters would use with their own twist and because the game was highly stylized so I had a lot of room to make the effect.

Before making this particular effect, I had already created and tested a few different ideas for this ability, but they never felt strong enough. I tested different combination that had texture sheet animation of a hand-drawn fire animation what would be rendered on a spiral mesh that wrapped around the character. This created a very good-looking effect, but it was lacking the powerfulness that it needed. I started looking for references from various sources. I knew games like Overwatch had very good visual effects, so I started there. I had played a lot of Overwatch, so I knew that characters like Reinhardt and Doomfist had effects that looked very powerful as they shattered the ground with their attacks. This gave me the idea that I could use the same idea for our game. I wanted to make the ground brake underneath the character when the ultimate ability was activated.

Picture 11. Doomfists ultimate ability shattering the ground.

The Doomfist type effect could have been made quite easily by just drawing similar cracks and leaving the uncracked parts empty and making a normal map to give it a little depth. I experimented with a couple of different hand-drawn sprites, but it was hard to bring the fire element to this. I also wanted the effect to look like the whole ground was blown into pieces and some lava to be visible from between the cracks. This gave me the idea that this could be achieved with a mesh.

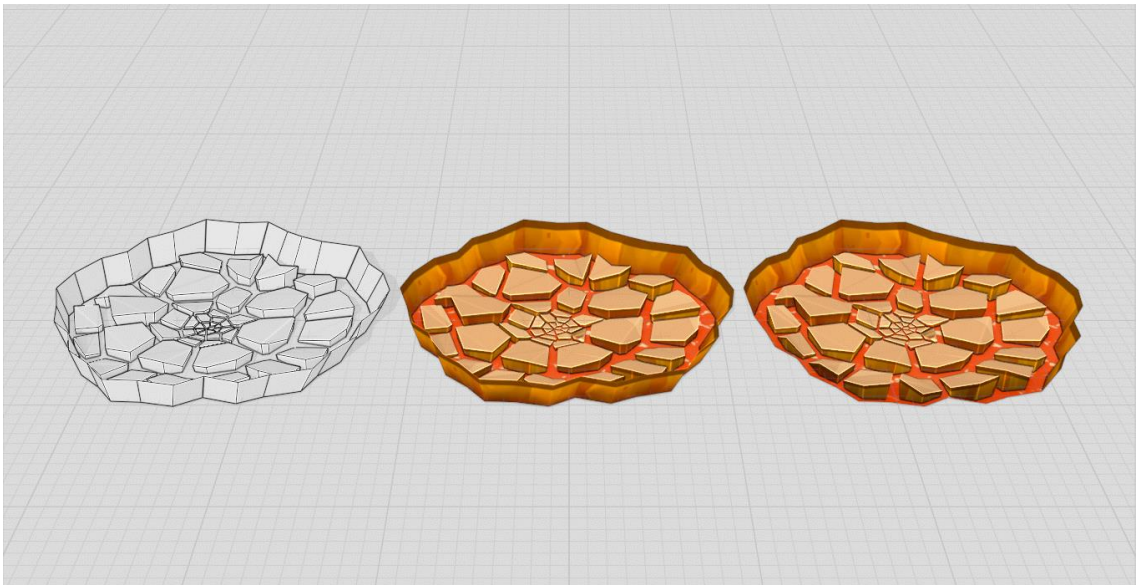
5.2 Creating the Effect

I opened Blender and started to make a shape that reminded a hole. I started with a single plane and subdivided it a couple of times to give it more roundness but still keeping some edges sharp. Blender has a built-in addon called Cell Fracture that fractures a mesh into smaller pieces. I adjusted the settings, so the fracturing made smaller pieces to the center of the mesh and bigger on the outside. I then scaled them, so they had gaps between them and extruded them down. I lowered the center parts to make it look more like an impact crater and because the character would stand there. After this, I simply added a cylinder around the pieces and removed the top face so they would be

visible. Aligning the edges of the cylinder to fit around the fractured mesh made it look like a hole that was a result of an impact.

Now there was a problem of how to blend it to the game scene's ground because this would be on top of the ground but at the same time, it needed to look like it's under it. Because the game has a camera behind the character and game engines normally render only one side of a mesh, I knew I could use this as an advantage. This is called backface culling. For example, a mesh that only has one face is visible from one side. I made the faces of the cylinder to be visible on the inside so if you look at it at an angle only the furthest faces would be visible giving it a feeling that is actually a hole. I put a camera to my blender scene to mimic the in-game camera and carefully adjusted all the faces to tilt a little forward to give even more depth to it.

I created the lava texture in GIMP by making an orange background and added some dark and white spots for contrast and variation. I made the texture seamless with a tool inside GIMP so I could use the same texture in the future. For the ground, I used an existing cliff texture and colored it to have a red tint close to the lava and a darker color on top to make it feel natural and blend better. Because this was a mobile game, I had to take into account that the effect wouldn't cast or receive any shadows for optimization reasons. This is why I baked an ambient occlusion map for the whole mesh and added it on top of the textures to give it some shadowing. The top parts of the mesh I separated and colored them to match the texture that was used in our game scene. I separated the parts because the same effect would be used in different maps with different textures. This way it would be easy to change the material for different map types.



Picture 12. From left to right: 3D mesh, 3D mesh with textures and 3D mesh with textures and backface culling.

A way couldn't find a way to spawn the ground mesh that made it look natural. I tried scaling and fading it in quickly, but it didn't look convincing. I decided to have a blast of clouds to spawn at first so they could block the view for spawning the mesh. I created the clouds in Blender using metaballs and converted them to meshes.

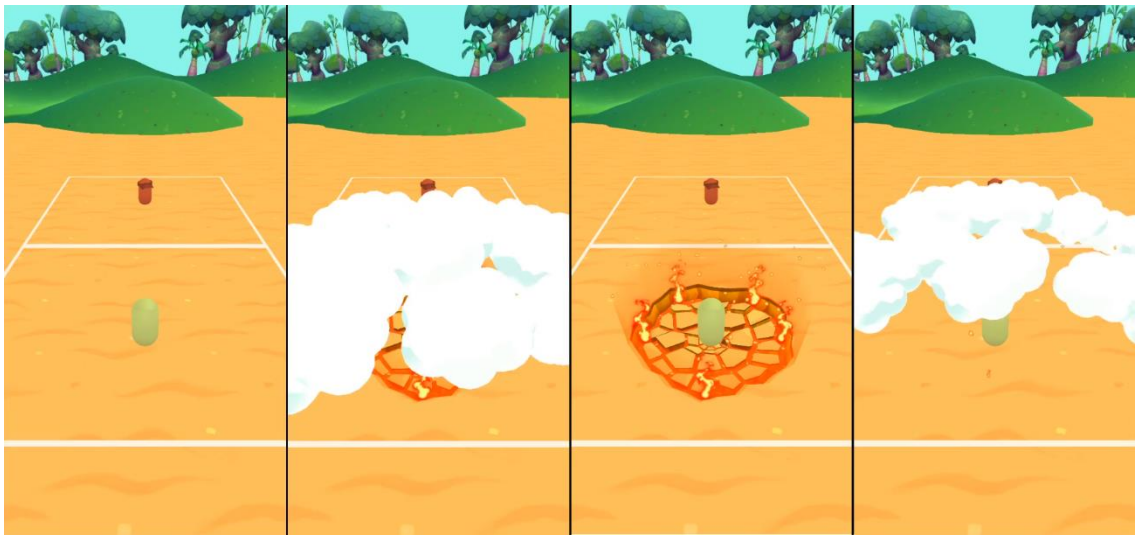
The lava wasn't enough by itself to bring out the fire element to the effect. It needed to have flames. Gladly we had a variety of sprite sheet animation of flames available. I added five separate flame animation sheets around the ground mesh. The problem was that they looked identical because they all used the same animation. I fixed this by giving all of them a slightly different simulation speed from the particle systems main module. The flames added movement and liveliness to the effect. The flames added movement and visualized the elemental aspect of the effect, but I wanted to add even more movement. I wanted the lava under the cracked ground to feel like its alive. At first, I thought the easiest way to approach this was to make a simple shader that moves the texture in the texture space but found an easier solution. I wrote a simple C# script that does the exact same but is usable with any existing shader. This was a better solution because now I could move the texture in any object I wanted without creating a new shader for different situations. I also had to separate the lava from the mesh so the lava

would be the only texture moved. The script has changeable parameters for the direction and speed it moves.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class TexturePanner : MonoBehaviour {

            public float ScrollX = 0.3f;
            public float ScrollY = 0.0f;

    private Renderer rendererToChange;
    private bool hasProperty = false;

    private void Start()
    {
        this.rendererToChange = GetComponent<Renderer>();

        if (this.rendererToChange != null && this.rendererToChange.material != null)
        {
            this.hasProperty = this.rendererToChange.material.HasProperty("_MainTex");
        }
    }
            private void Update()
            {
            if (this.rendererToChange != null && this.hasProperty)
        {
```

```
            float OffsetX = Time.time * ScrollX;

            float OffsetY = Time.time * ScrollY;

            rendererToChange.material.mainTextureOffset = new Vector2(OffsetX,
            OffsetY);

        }

    }

}
```

For the final touch, I added a cylinder mesh to the outlines of the effect and textured it with a gradient texture that fades out. It gives the feeling that the lava is burning hot.



Picture 13. The final effect.

6 TESTING

6.1 Testing method

The finished effect needed to be tested. The Game Experience Questionnaire (GEQ) is a widely used and very comprehensive method to test a game where a tester answers to a series of statements about the game. I felt I couldn't get precise data for the effect itself with the GEQ because it measures the entire gameplay and I only needed to test a small portion of it. Since visual effects are a fairly new field in the gaming industry, there is not much literature on the subject, and I couldn't find any general method for testing them, so I had to create one. I took inspiration from the GEQ but made it much more focused on the effect and modified the statements. The GEQ measures a variety of things like competence and flow but I decided to focus only on negative affect and positive affect. The purpose of this thesis was to create a clean and clear visual effect that supports the gameplay, so the statements were created to answer if those goals were achieved.

I had a group of 11 people to test the game. The testers were asked to play at least until they could use the Ultimate Ability. After playing the game the testers were asked to answer 14 statements about the ultimate ability in two parts.

In first part testers answered the statements 1-8 on the following scale and scoring;

not at all	slightly	moderately	fairly	extremely
0	1	2	3	4

and in the second part the statements 9-16 on the following scale and scoring:

strongly disagree	disagree	either agree nor disagree	agree	strongly agree
0	1	2	3	4

The statements were following:

Part 1. When I pressed the Ultimate Ability button…

1. I was confused
2. I felt powerful
3. I felt underwhelmed
4. I felt skillful
5. I felt excited
6. I understood what happened
7. I was annoyed
8. I felt aesthetically pleased

Part 2. The Ultimate Ability...

9. Didn’t look good
10. Cluttered the screen
11. Fit the games graphics
12. Looked good
13. Gave me a bad mood
14. Disrupted the gameplay
15. Looked out of place
16. Improved the gameplay

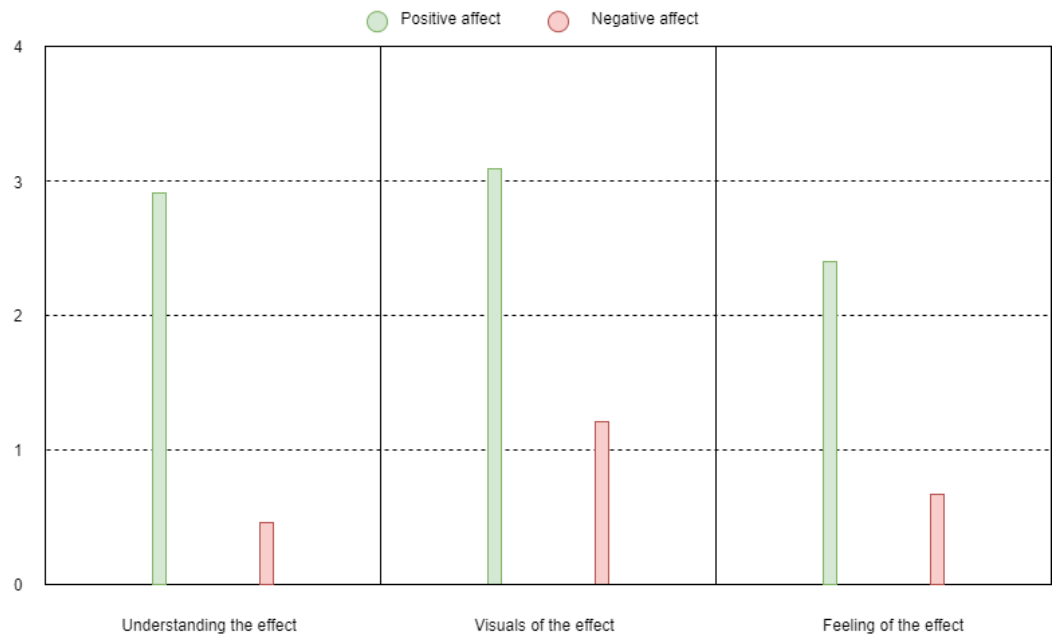
The statements are divided into three different themes. All three themes have an equal amount of negative and positive statements to measure the affect:

- Understanding the effect
  - Positive **6, 16**
  - Negative **1, 7**

- Visuals of the effect
  - Positive **8, 11, 12**
  - Negative **9, 10, 15**

- Feeling
  - Positive **2, 4, 5**
  - Negative **3, 13, 14**

6.2 Results



Picture 14. Testing results.

The results were reasonably good but leaves room to improve. Overall score for the visual effect was 2,83 out of 4 for the positive affect and 0,78 out of 4 for the negative affect. The positive affect is noticeably higher than the negative affect in all three themes. Testers seemed to understand what was happening when the ultimate ability was triggered and what the effect portrayed. Visually the effect was considered to look good and fit the games graphics, but the higher negative effect seems to point that further work to make the effect clearer is needed in the future. The feeling of the effect can be also considered as good but needs some improvement. The testers didn't seem to feel underwhelmed but neither powerful enough that I had hoped for. In further tests it would be useful to get feedback as a free word format to have a better understanding behind the reasons.

Due to a relatively small testing group the results can be considered as indicative and portrait 11 persons subjective view. Visual effects are mostly considered as art and therefore based on subjective opinions. This study is relevant because individual

subjective opinions are valuable to study but bigger testing groups are needed to get a more generalized result.

The testing method didn’t consider the players opinion about the whole game which can affect the results. The purpose of this thesis wasn’t to study the whole game due to which I didn’t need information about how the whole game made them feel. Therefore I felt it was important to focus on the effect it self. If I had studied the game I would have ended up with huge amount of data that wouldn’t help me understand is the visual effect good.

7 CONCLUSIONS

There are many tools and techniques required to master in order to work as a VFX artist. As game engines and the devices used for gaming are constantly improving the skills needed are also constantly increasing. Understanding how particle systems work is a good starting point but knowing how to use 3D modeling and photo manipulation software are the minimum skills required to work as a VFX artist. As VFX artists often don't have much art direction the planning and composing also fall under the artist's responsibility.

The goal of this thesis was to create a clean visual effect for a mobile game and improve my own skills and widen the knowledge of techniques as a VFX artist. My hope is also that this thesis is helpful for anyone wanting to work as a VFX artist in the game industry as there isn't much literature on the subject due to the field being quite young. This thesis should give a good understanding of the tools and techniques and where to start. Writing this thesis gave me a wider perspective on the game industry and visual effects in general.

For the effect I created, I went through all the processes from planning to implementation and explained the reasons for the choices I made. The effect created used a variety of different techniques from texture sheet animations to 3D meshes and some coding. It is clearly visible on a small mobile screen and it's optimized well. It doesn't clutter the screen nor disrupt the gameplay but supports it as a visual effect should. Visual effects have become a passion for me due to the combination of technical and visual aspects and I hope to make visual effects for my job in the future.

REFERENCES

Alisavakis, H. 2017. My take on shaders: Dissolve shader. Referenced 18.11.2020. https://halisavakis.com/my-take-on-shaders-dissolve-shader/

Chamberlain, H. Keyser, J. 2017. Visual Effects Bootcamp: Artistic Principles of VFX. Referenced 3.11.2020 https://www.gdcvault.com/play/1023943/Visual-Effects-Bootcamp-Artistic-Principles

GentleGaming. 2019. Unity Shadergraph Dissolve Shader. Referenced 18.11.2020 https://www.youtube.com/watch?v=kX39KdgEyrY

Guerrette, K. 2016. The Desired Effect: How Visual Effects Are Essential to Video Game Storytelling. Referenced 9.11.2020 https://www.youtube.com/watch?v=rBvXfJqjdc4

Grissom, S. 2018. Real-Time VFX: A Visual Language Spectrum. Referenced 3.11.2020 https://www.gdcvault.com/play/1025230/Real-Time-VFX-A-Visual

McDonald, A. 2020. Most Popular Software 2020 - VFX, Animation and Games packages you need to learn. Referenced 16.11.2020 https://discover.therookies.co/2020/02/28/most-popular-software-2020-vfx-animation-and-games-packages-you-need-to-learn/

Oberbeck, J. 2017. FMX2017 Technical Directing Special: Real-time Volumetric Cloud Rendering. Referenced 21.11.2020 https://www.youtube.com/watch?v=8OrvIQUFptA

OMEGZKI GAMING STYLE. 2020. Diablo Immortal: Class Gameplay and Abilities. Referenced 18.11.2020 https://www.youtube.com/watch?v=WNQsz2MsSFU

theRadBrad. 2017. THE EVIL WITHIN 2 Walkthrough Gameplay Part 1 - Kidman (PS4 Pro). Referenced 12.11.2020 https://www.youtube.com/watch?v=QjvAjNZSgjI

Thorn, A. 2015. Unity Animation Essentials. Packt Publishing Ltd.

Unity. 2020. Particle System. Referenced 4.11.2020 https://docs.unity3d.com/Manual/class-ParticleSystem.html

Unity. 2020. Practical guide to optimization for mobiles. Referenced 20.11.2020 https://docs.unity3d.com/Manual/MobileOptimizationPracticalGuide.html

Valve Developer Community. 2020. Particle Editor. Referenced 12.11.2020 https://developer.valvesoftware.com/wiki/Particle_Editor

Encz, B. 2015. 5 Best Practices Beginner Mobile Game Developers Must Know. Referenced 20.11.2020 https://www.codementor.io/@ben-g/mobile-ios-game-development-best-practices-ajfy1ck1k