Bachelor’s Thesis

Information Technology

Game Technology

2018

Jami Aho

REWARDING PLAYERS WITH AUDIOVISUAL CUES

– Giving feedback through visual effects.

TURKU AMK

TURKU UNIVERSITY OF APPLIED SCIENCES

BACHELOR'S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information Technology

2018 | 52 pages

INSTRUCTOR: Principal Lecturer Mika Luimula, Adj.Prof.

Jami Aho

REWARDING PLAYERS WITH AUDIOVISUAL CUES

- Giving feedback through visual effects

The objective of this thesis is to have a look into why rewarding and guiding players with audiovisual feedback is important. To see what means real-time visual artists have, to accomplish this and how they have been used in production. This is done by studying the articles, presentations and games published by various developers and artists.

Many developers are using the word "juice", which is loosely described as immediate feedback that is constantly and generously given to the player. The visual aspects of giving feedback and pleasure from interaction is essentially what this thesis revolves around. The purpose of this study was to get a deeper understanding of the fundamental reasons for adding visual effects and what areas of expertise are needed from the creator.

Know-how from this study has been utilized to craft effects for the game Moomin: Match and Explore. Well-fitting and generous visual effects have been created to improve user experience and game feel. The game has been released for Android and iOS mobile devices.

KEYWORDS:

Audiovisual, feedback, rewarding, juice, visual effects, game

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tietotekniikka

2018 | 52 sivua

OHJAAJA: Yliopettaja Mika Luimula, dos.

Jami Aho

PELAAJIEN PALKITSEMINEN AUDIOVISUAALISILLA VIHJEILLÄ

- Palautteen antaminen visuaalisten tehosteiden kautta

Tämän opinnäytetyön tavoitteena on tutkia, miksi audiovisuaalisen palautteen antaminen on tärkeää pelaajaa palkittaessa ja ohjattaessa. Työn tarkoituksena on selvittää, millaisia visuaalisen palautteen keinoja graafikolla on reaaliaikaisia tehosteita käyttäessä. Lopuksi tarkastellaan, miten näitä tehosteita on käytetty omassa pelituotannossa. Tämä on tehty tutkimalla muiden kehittäjien ja artistien julkaisemia pelejä, artikkeleita ja esityksiä.

Monet pelinkehittäjät ovat alkaneet käyttämään sanaa ”juice”. Tällä tarkoitetaan välitöntä, jatkuvaa ja runsasta palautteen antamista pelaajan reaaliaikaisista interaktioista. Opinnäytetyö pyrkii keskittymään pelaajalle annettavaan visuaaliseen palautteeseen: palkintoon ja nautintoon. Työn tarkoituksena on saada syvällisempi ymmärrys peleihin lisättävien visuaalisten tehosteiden tavoitteista ja tekijän kannalta tärkeimmistä osaamisalueista.

Erilaisia tehosteita ja muuta työn aikana tutkittua materiaalia on hyödynnetty Moomin: Match and Explore -peliin tehtyihin visuaalisiin tehosteisiin. Android- ja iOS-mobiililaitteille julkaistussa pelissä on tarjottu anteliaasti peliin sopivia visuaalisia tehosteita, joiden avulla on haluttu parantaa pelikokemusta, -tuntumaa ja -nautintoa.

ASIASANAT:

Audiovisuaalinen, palaute, palkitseminen, visuaaliset tehosteet, peli

CONTENTS

FIGURES

GLOSSARY

Polish	To make and improve the visual and auditory elements to be more understandable and fitting for their meant environment (Swink 2009).
Juice	Generous audiovisual feedback cues for the player to feel satisfaction and reward (Gray et al., 2005).
UV map	When a 3D models coordinates are flattened onto a 2D image (Bech-Yagher 2017).
VFX	Visual Effects
UI	User Interface

INTRODUCTION

Visual effects that work in real-time are used in essentially every game nowadays. Some of these visual effects attempt to give players feedback from their actions. These effects might be visual points of interest, audiovisual rewards or a replication of a real-world event, e.g. a button press. These effects, try to guide and give joy to the player and make the application or game feel complete.

Adding positive feedback effects that are spawned from the player's interaction have become more discussed by game developers in the past years. There were few findings in literature for this subject, but developer presentations and online articles had more information. However, some additional literature sources are being included for the parts that go through some older and more discussed themes like human behavior and game development phases. Software documentation and tutorials are also being referenced in several places.

The immediate feedback that is constantly and generously given to the player is called "juice" by many of the writers and developers introduced in this study. Interest into this area of game development has grown, but the information seems to be scattered all around. Real-time visual effects also require knowledge in different areas of expertise and accumulating that experience will take time. This thesis attempts to tie down some of the most used visual effects for giving feedback about the game's progression, why they work and how they have been used in a project.

While artists need countless hours to be put into practice to become successful, could we still have a look behind the curtains of intuition? What makes effects look good, generates emotions and feels fun? What should the developer focus on while learning and creating new effects? These are all questions wanting to be cleared.

In chapter 1. The Importance of Giving Feedback, the aim is to have a look into what makes a good feedback effect, how others have done it, what practices are common and what links them together. The second chapter will try to define the elements that can be used to build some of the many visual effects. This part will also try to identify what parts of our game we can emphasize with added effects and some of the guidelines we should remember.

In the last chapter we will see how some of these effects are being used in the mobile game Moomin: Match and Explore. This game is developed by Snowfall Ltd. The last chapter will also include author's thoughts, conclusions and things learned, during this process.

1. THE IMPORTANCE OF GIVING FEEDBACK

1.1 Fun in games

Fun is defined as pleasure, which means that endorphins are released and a good feeling in your brain emerge. Pleasure can come from many different things such as leisure activities, work, social interactions and surprisingly small normal day to day activities. In games the good feeling can come from a plethora of tasks, achievements, aesthetics and “Aha!” moments. Raph Koster (2013) believes that learning something is the most important fun moment, since learning and mastering has always been important for survivability. (Koster 2013, 40.)

Fun is a very personal concept that varies from person to person. What is fun for one can be extremely boring for another. It applies to all kinds of activities and in video games there are multiple subcategories and many different experiences that have their own fan groups. (Dillon 2010, 15.)

What first was fun can also become boring. The lack of stimuli from repetitive or dull tasks can be off-putting and if a story does not have a good progression and interesting plot points we might start to question our reasons to continue. So, even if a game presents something new and exciting, it can quickly become boring, if it fails to continue delivering us new data to process. Too much information from a game can also go to the wrong direction and give the player a sensory overload. Game developers have to keep balancing to offer an enjoyable experience with enough fun that is not too cluttered with new mechanics to learn. (Koster 2013, 42.)

1.2 Polish

Polishing in the real world means creating a smooth and shiny surface. The meaning is quite similar in game development where developers want to add and improve the visual and auditory aesthetics of the game. It is an essential part of the gaming experience and there are almost no games that don’t have any polish in them. The question of what game polish means is hard to answer, but developers have a similar idea of it.

J. Matthew Zoss (2009) has gathered thoughts from multiple developers in his article “The Art of Game Polish: Developers Speak”. Generally, they think that a polished game is something that does not pull the player out of the experience by having features that seem out of place. All game productions should also have a well-planned timetable and a slot reserved for polishing and finishing the game. The polishing phase often consist of making the game flow and feel better, adding new elements that enhance the existing mechanics of the game and fixing everything to truly work as desired. (Zoss 2009.)

In Steve Swink’s book Game Feel (2009), he explains polish in games to be one of the three building blocks for game feel. The other two are real-time control and simulated space. Real-time control is when a player and a computer communicate in a closed loop. This is when a player is for example controlling a character or driving a car and the computer translates player interaction to real-time movement on the screen. Simulated space is the surrounding virtual space that the player can interact with. The third building block, polish is something that will not change how the game works but will enhance the interactions and give feedback for the player. Polish effects are artificial audiovisual cues for the player to understand what they are doing in the game. Steve Swink describes game feel like this: “Real-time control of virtual objects in a simulated space, with interactions emphasized by polish.”. But continues to explain that not all games have these building blocks for game feel. (Swink 2009, 5.)

For a game that has real-time control of virtual objects in a simulated space, the polish can be the animations of a virtual character or the dust particles that emerges from running on the desert sand. When a player hits enemies or just bare ground it should give a certain effect as feedback for the player. It’s important to note that the colliders or physical simulation that detect the hit on the enemy are not part of the polish effect, but they will observe if the hit was successful and output a polish effect. It might be combination of animations, sound effects, particle effects or some other graphical effects. Functionality in the game would still work the same way even if the polish was removed (Swink 2009, 151).

Let’s imagine a normal sized human character with a set of attack animations that looked quite realistic. The animations would sell the character, its mass and the velocity of movement for the general audience, since they look like something that works the same as in real life. This will also apply on more stylized characters as long as their movement is on par with their displayed mass, size and look. But if the virtual character was sized up ten times and the animations were left untouched, it would just look weird and unfitting. This would be badly polished and to iterate it further we would have to change the animations to look like they belong to a giant. To polish is to make and further refine animations, game objects, environments, graphical effects and shiny armors, so that the audience perceives them as authentic, united and fitting for their meant space.

1.3 Juice

Many developers have started using the word “juice” in their talks and articles. Juice as a term originates from Gamasutra’s (2005) online article “How to Prototype a Game in Under 7 Days”. They describe juiciness as immediate feedback that is constantly and generously given to the player. A juicy feedback effect can be a flash, wiggle, bounce, splatter, sound or any other audiovisual response from an action that makes the game feel more alive and fun. It will help to communicate how well the player is doing and empower the gameplay. (Gray et al., 2005.)

Articles, presentations and other information have then started to emerge. Many of the developers in this thesis define juice by showing examples and sharing their thoughts. (Jonasson & Purho 2012, Brown 2016, Nijman 2013, Berbece 2015, Turner 2015.) Some polish will be considered as juice, as seen in figure 1.

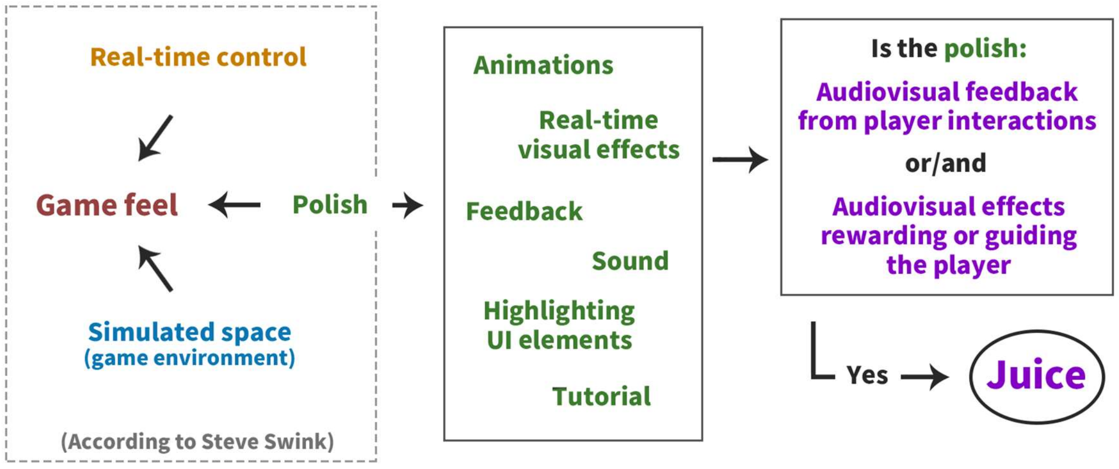

Figure 1. What is juice?

Juice is also polish, but it is limited to the immediate feedback that the player gets from the game. The animations of a slow-moving giant are considered as polish but the effects from its footstep can be thought as both juice and polish. Figure 2 demonstrates the difference that added juice can make. Particles, explosions and lights can be seen in the image. But, animations, sounds, screenshake, haptic feedback, small pauses and flashing lights that cannot be presented in an image, also play a big role (Forestié, Gameloft Montreal 2018).

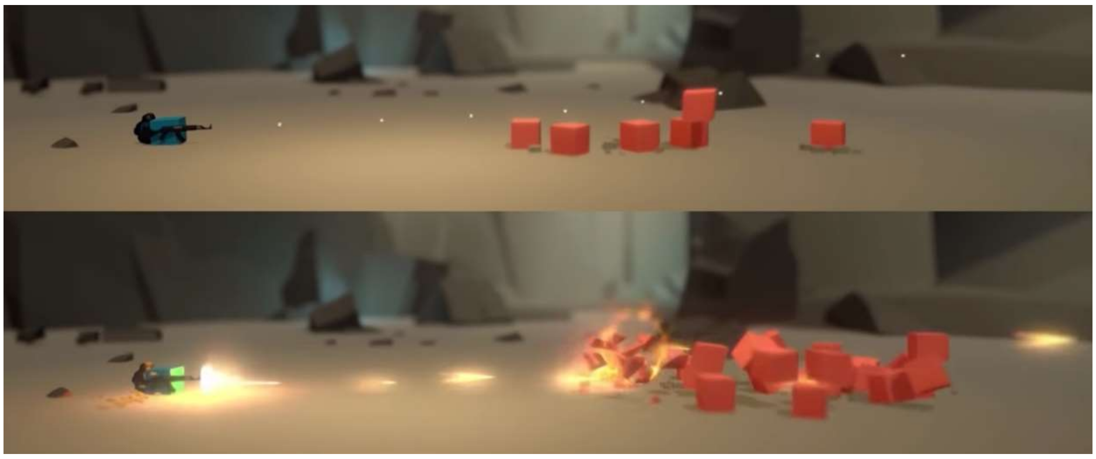

Figure 2. With and without juice (Forestié, Gameloft Montreal, 2018)

Different effects and animations are usually combined to give the player a better and more interesting overview of the situation. Jonasson and Purho (2012) describe juice to be the stuff that makes the game feel good to interact with. In their talk “Juice it or lose it”, a breakout-like game (Figure 3) is used to showcase different ways to add juiciness. Movement is made to look more interesting by interpolating the object’s transitions. This is used to remove sharp and harsh movement. The ball scales bigger, flashes white and wiggles when it’s hitting objects. It also makes the screen shake and leaves a trail. Sound effects and music are also shown to have a massive significance in how the game feels. Many tricks are used to accomplish a much more entertaining and satisfying experience without changing the underlying simulation. (Jonasson & Purho 2012.)

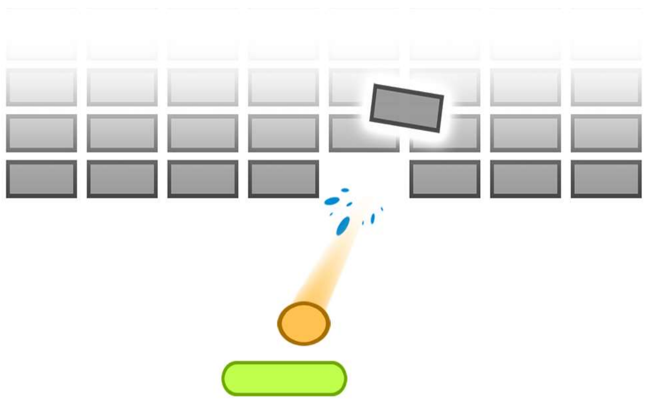

Figure 3. What Breakout-like games look like

But juice doesn't have to be fast and aggressive. In Lisa Brown's (2016) talk "The Nuance of Juice", she explains how she uses small details, calming light effects, pauses and other features that make her game more harmonious. It is important for the effects to not feel forced and many times things will seem weird or out of place at first, but by tweaking they become more fitting for the experience. Brown also notes that purpose should always come first, and we can help ourselves by asking questions like: Why are we juicing it? What do we want to achieve with it? What should it look like? For Brown's game, where the player catches firebugs, one of the aims was to have the screen less static and introduce subtle movement to indicate where player's attention should focus. Feedback juice for the game became quite minimal and had the desired calming meditative effect on players testing it. (Brown 2016.)

1.4 Emotions and patterns

One way to generate emotions from a player is using visceral experiences. With appearances, sounds and other ways to deliver the feedback developers can convey certain feelings. Screeching tires and flying dust when driving close to the edge, collecting precious sparkling gems or hearing meaningful sounds or music can all deliver emotions. (Lazzaro 2004.)

These experiences often present themselves naturally for the player and thus are quite easy to experience. Juicy effects and minor polishes that might seem insignificant oftentimes make the essential feeling that the developers so yearn for. Brown notes that game development students have increased their efforts in using juice over the past years and concludes her talk that students should have more emphasis on player’s feelings and focus the juice on those feelings that you want to target in your game (Brown 2016).

Aesthetic appreciation can come from recognizing patterns and those surprising moments of familiarity can produce enjoyment for the player. Recognizing a pattern can give you chills, whether it is from a story suddenly making sense, a memorable piece of music starting to play or understanding the pattern to solve a puzzle (Koster 2013, 94,146). Graphical effects like glittering and flashy beams can be a sign that a reward is ready to be picked up or that it is about to pop on the screen. Players will remember these things from having been exposed and made to memorize the patterns during their current and previous games. “Best thing we had going for us was our intelligence, especially our gift for pattern recognition, sharpened over eons of evolution." (Tyson 2015).

1.5 Colors and sounds

Game designers can influence player emotions with colors (Joosten et al., 2010). “Color is a powerful communication tool and can be used to signal action, influence mood, and even influence physiological reactions.” (Cherry 2018).

In Nintendo’s 1998 game The Legend of Zelda: Ocarina of time we have the helpful fairy called Navi, who is changing colors depending on what is close by. Navi will have a blue glow when there is a possibility to communicate with someone, a yellow glow to warn of an enemy or a green glow when there is something of interest nearby. Navi also has sound effects to contribute on getting player’s attention when she has something to say. (Nintendo 1998.)

“Most blues convey a sense of trust, loyalty, cleanliness, and understanding.” (colormatters 2018).

While people usually respond to certain colors in similar ways the meaning of the color can be changed. In Far Cry 4, when the player first enters the Shangri-La area within the game, the scenery turns to a mix of red, orange and gold (Figure 4). There isn’t much but nature, peace and old ruins at the start, but the world gradually evolves toward a more blue and hostile experience. Blue particle systems splatter when cutting down the bluish enemies that occupy the land of Shangri-La. Developers wanted to emphasize the presence of enemies in the closing missions by coating the world in blue as seen on the right side of Figure 4. (Cook 2015.)

Figure 4. Screenshots of Shangri-La within Far Cry 4 (Ubisoft 2014)

Colors and themed music can be used as training to recognize patterns in a variety of situations (Koster 2013, 73). Variations in game music can produce different gut feelings like intimidation, happiness and many more. Theme songs often also try to describe the surroundings for the player with the help of tempo, dynamics, style, different instruments and sounds mixed from the environment itself.

Sound effects play a huge part in conveying information, emotions and harmony with the rest of the experience. In Joonas Turner’s (2015) presentation “Oh My! That Sound Made the Game Feel Better!” he explains that it’s important for developers to understand what to juice in video games and to keep the sounds cohesive. It might be a soldier’s movement and action sounds or the card shuffling sounds in a card game. Audio cues can help the players notice what is happening without relying on visuals. (Turner 2015.)

Sound effects are left a bit short in this thesis, but they play a big role in delivering both positive and negative feedback as the game developer has intended.

1.6 Reward from action and goals

1.6.1 Intrinsic motivation

Intrinsic motivation is when we are doing something because we enjoy the activity itself. It can be fun, challenging or something else, but there aren’t any external goals, pressures or rewards affecting the motivation. (Ryan & Deci, 2000:56.)

In a role-playing game where the player completes quests by fighting their way through challenges, the intrinsic motivation often comes from the actions and fighting. In Dragon Age games, you can carefully plan out your combat phase by using the pause mechanic to strategically choose the next skills to be used by your characters. Diablo 3 has a fast gameplay, compared to the slow-paced Dragon Age, where players compete to see whose character can hack and slash the content faster.

In both games, the combat is pleasing and fun for many. In addition, there is the story to follow, puzzles to crack and plenty of other challenges. These can be counted as intrinsic motivators and they all can be made more interesting by adding a layer of effects on top.

1.6.2 Extrinsic motivation

"Extrinsic motivation pushes you to do (or avoid) something because of an external reward or punishment." (Zichermann, 2011).

One might not like the work or study they do, but with the money, grades, social interactions and social status as a reward and motivator they are able to keep doing it. After starting school, children tend to have their intrinsic motivation get weaker each passing year and extrinsic motivation is there to help the journey ahead. (Ryan & Deci 2000, 60.)

In games, players usually work towards multiple goals. In the Dragon Age and Diablo games extrinsic motivators can be leveling up your characters, getting crafting materials for better equipment and doing a variety of tasks to get rewards or to get to the next step. When these align with entertaining intrinsic motivators the outcome can be very addictive. By adding highlights and effects where extrinsic motivators are shown, players are given more feedback, reward and possibly an unconscious goal to work towards.

1.7 Usability experience in mobile games

Mobile games for operating systems like Android and iOS have become extremely popular after Google and Apple opened their app markets for developers. Especially games and applications for smartphones have more user interface real-estate, since phones generally rely on touch input from the screen rather than a keyboard or some external controller.

In Nintendo’s mobile game Super Mario Run there are animations here and there. Speech bubbles are shaking, buttons jump upwards when pressed and there is a screen transition effect between almost every scene and menu. After winning a level, Mario jumps up and shouts “Mario Time!”, confetti bursts on the screen and the player is taken to see their score. Time, coins and other data appear on their place with different animations, particle systems and sound effects. Other mobile games also use a lot of animations and effects to spice up their user interface.

Nielsen’s 10 usability heuristics for User Interface Design are good to remember when designing user interfaces, but some of them can be adapted to the effects we place in our games. Effects can be easier to understand if they match something that is in the real world and it’s good to keep things consistent and in some cases similar as in other games. This way players might recognize something familiar. Effects can also help to achieve an aesthetic and minimalist design by giving feedback that doesn’t have to be in the user interface. Feedback effects will also confirm actions and guide the player. (Nielsen 1995.)

2. WAYS TO IMPLEMENT JUICE

2.1 Key elements for success

There are many ways to juice your game but making something look and sound good can be difficult and often require help from team members. With great audiovisual possibilities comes great responsibilities and a steep learning curve. Technical software knowledge might be needed in image editing, 2D painting, 3D modeling, 3D sculpting, shader creation, graphics programming, implementation programming, game engines, particle systems, textures, asset optimization, UV editing and so forth, depending on the needs of the game and team (Guerrette 2018, Jevremović 2018, de Laat 2018).

A real-time visual effect artist should be familiar with some animation principles to ensure interesting movement and to understand anticipation, climax, dissipation and good timing in general (Keyser, Riot Games 2018). Awareness and communication between the team is also vital to success in making a believable visual effect that works in its environment. Last, but not least, one should have down the fundamentals in art and hopefully be able to concept and visualize their effects beforehand (de Laat 2018).

Being such a broad subject, this section will not delve so much into explaining everything to its core but expects the reader to know the basics in game development. There will also be very little comparison between different tools. The bulk of the text will be about the different ways to combine and infuse juicy effects into a production and why it is important.

2.2 Visual Effects (VFX)

“Great VFX are one of the most direct ways to add visual richness and production value to your game.” (Kuipers 2016). A real-time visual effects artist’s job is to breathe life into the game world that otherwise might seem static and motionless (Guerrette 2018).

Francisco Ordóñez (2017), a senior VFX artist talks about two major types of tasks, gameplay effects and environmental effects. How much effects are needed in each category differs from game to game. In the hack and slash game Castlevania, 90% of the visual effects were in character or enemy powers, attacks and magic spells. Ordóñez continues that these kind of tasks, really need a good understanding of the game, since they affect the gameplay a lot. (Ordóñez 2017.)

Riot Games also emphasizes the importance of well-fitting visual effects. Their VFX goals for their hugely successful game League of Legends are to:

-Provide visual clarity for gameplay

-Minimize visual clutter

-Deliver visual effects that promote a champion’s themes

-Create effects that surprise and delight players

(Riot Games 2017, 4)

Visual effects for a competitive online game need to be spot on to communicate gameplay clearly and accurately. In Figure 5, the primary elements (blue circles) of the effects are clearly visualized to highlight the focal points and range of the champion’s skills. The secondary points (small yellow triangles) enhance the overall visual appeal by adding thematic values and color hues, value and saturation. (Riot Games 2017.)

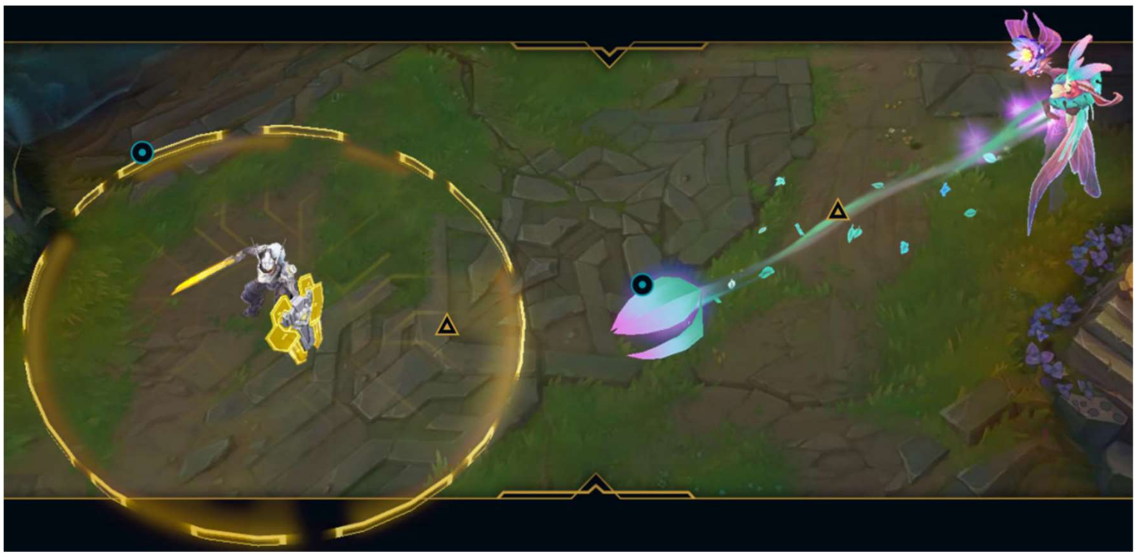

Figure 5. Primary and secondary elements of VFX in League of Legends (Riot Games 2017)

Whether it’s designing 2D, 3D, VFX or other visuals, it is always crucial that it fits in its desired environment. The spells used in Figure 5 not only match the colors of their casters, but also their thematic ideals. With a good color value range, effects can stand out positively from the environment. Different color palettes and variations can also help to define different magic categories in games. For example, a combination for fire is usually something like red, orange and yellow, but by adding purple and pink to that color-palette it becomes celestial magic in League of Legends. Healing, wind, darkness or arcane spells all have their own interesting style of color and we can further polish the magic-like effects by adding illumination to bleed from the primary shapes as seen in Figure 5.

VFX can also be something simple like a small flash highlighting the experience bar whenever it is increased. It is good to remind players that extrinsic motivators are there and to give them feedback on how their adventure is going. Minor effects are sometimes almost unnoticeable, but it would still feel strange if they were missing. Subtle elements like this can take time and determination to achieve. (Brown 2016.)

2.2.1 Sprites and textures

Sprites are used everywhere, it is the easiest trick in the book. Sprites are 2D graphics objects where an image is rendered on two triangles as seen in Figure 6. User interfaces in games are normally made with sprites. Sprites are used as particles in particle systems and the 2D environments in games are usually built using sprites. (Unity 2018.)

Sprites used as effects are usually animated in some way after they are spawned in the scene and before they are deleted from the scene (Figure 6).

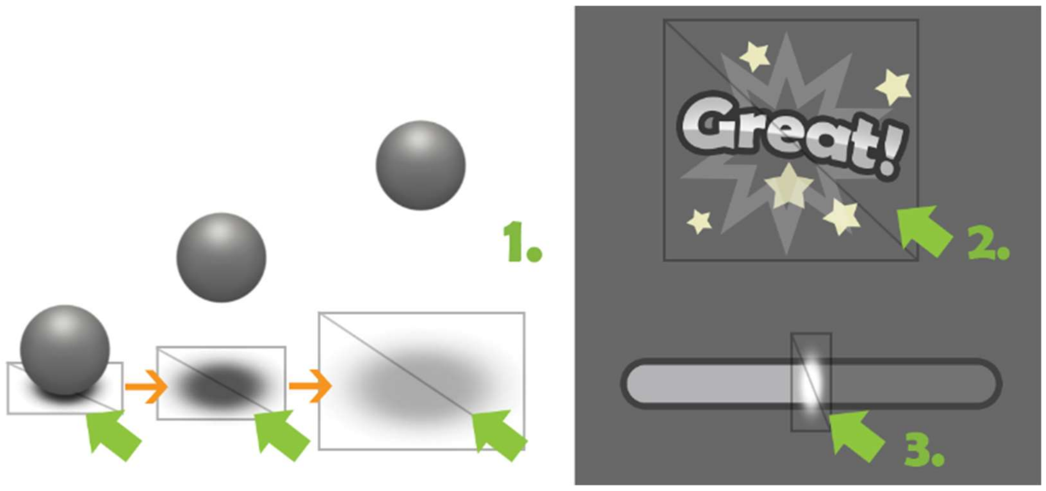

Figure 6. Simple use-cases for sprites

Sprites are a sub-category of textures which are basically 2D images. Textures can be laid on top of the triangles/polygons that meshes are made of. A mesh needs to be mapped to 2D-space to be able to correctly display a texture, this is called UV mapping, which we will take a better look at in chapter: 2.2.3.1 UV mapping and UV distortion. (IT Hare 2016.) Textures can be used for a variety of different things. For example, objects using physically-based rendering need multiple textures for different properties layered on top of the model (Marmoset 2015).

Our digital images are made from the three main colors: red, green and blue (RGB). Textures have a channel for each of them plus an optional channel for transparency called alpha (A). Each channel is presenting its corresponding color in greyscale and when added together they present the image in complete colors. The alpha channel is often used to mask parts of the image off. Artists can also pack different images in those greyscale channels and separate them with a shader to be used as various elements of the effect. Channel packing is used in games to avoid loading separate grayscale images, which will save memory. Although this will increase shader complexity and might in some cases increase draw calls. (Polycount wiki 2018.)

2.2.2 Shaders

Most visual effects done by shaders use vertex shaders or pixel shaders or both. Vertex shaders perform mathematical operations on the vertices that make up the triangles (polygons) which our 3D models are made of. Pixel shaders on the other hand calculate the pixel colors and lighting values for each pixel per each frame based on the values and textures given. (Nvidia 2018.)

Shaders are small programs that can do a task for one of the stages in the graphics rendering pipeline (Khronos 2015). Covering everything about shaders in this thesis would be impossible, but instead we will try to showcase the possibilities where shaders can be used in juicy visual effects.

2.2.3 Materials and shader creation

Textures for an object are combined in the shader, but the shader’s properties are often exposed inside the game engine’s editor. Premade all-around game engines such as

Unity and Unreal Engine can be used to do a variety of tasks related to games and other productions (Unity 2018, Epic Games 2018). It is possible to achieve similar end-result in both engines (Okulov 2018), but the features and their workflows are sometimes different, as we can see in their material creation process.

In Unity, all unique objects will have their own material with a user-selected shader that displays its material properties in the material editor (Unity 2018). In Unreal Engine 4, users will use visual scripting nodes to make materials for their objects. Each node contains a code snippet and this node-based solution will in the end generate a shader. (Epic Games 2018.)

Unreal Engine's material editor can also be used for creating shaders with movement and different effects like flames, smoke, water and so forth. This is also possible in Unity when using Unity's own Shader Graph (Figure 7) or purchasing Amplify Shader Editor or Shader Forge from the Unity Asset Store. All of these can generate shaders using a node-based approach.

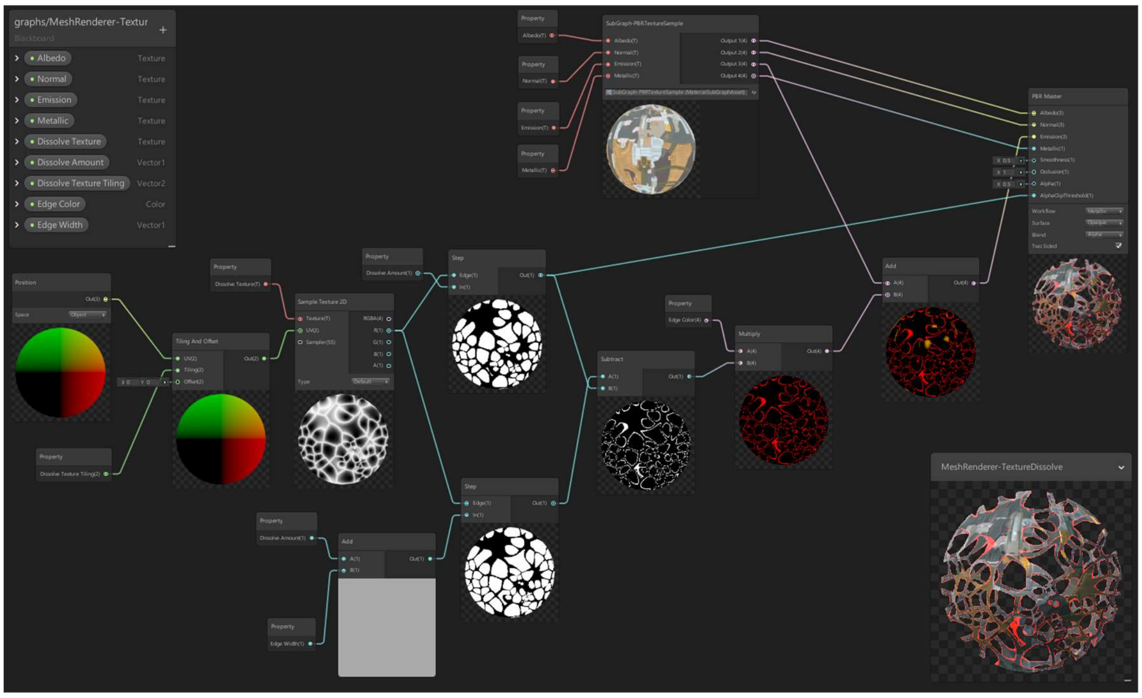

Figure 7. A screenshot of dissolve shader's nodes from the Shader Graph Example Library (Unity Technologies 2018)

Shaders made with the node-based approach is becoming adopted more and more each day since it is easier for artists and other developers to use. Similar popularity can be seen for visual scripting tools where game functions and code are compiled from nodes.

Everything made with a visual editor can also be made by writing just the lines of code, which was the only way shaders could be put together in past.

2.2.3.1 UV mapping and UV distortion

Shaders have many ways to alter or use the UVs and the UV space. UVs are the vertex positions of a mesh put onto a two-dimensional UV space (Autodesk 2018).

“This means the first thing you need to know about UV mapping is what UV space is. Based on a 0 to 1 grid, with 0.5 as the middle coordinates, a UV map consists of your 3D model's XYZ coordinates flattened into 2D UVW space – or tile, as it's called.” The letters UVW are used instead of XYZ to avoid confusion between the 3D and 2D space. (Bech-Yagher 2017.)

Distortion effects can add interesting movement to the texture. Simple distortions can be added mathematically or by editing the positions of the UV mapped XYZ points of the mesh.

“To support more interesting flows, we must somehow vary the flow vector across the surface of our material. The most straightforward way to do this is via a flow map.” (Flick 2017). The flow map in Figure 8 uses the R channel for the U component (x vector) and G channel for V component (y vector). Flick uses this flow map in his example to make water-like movement by moving it over the main texture, which in this case is the water surface texture.

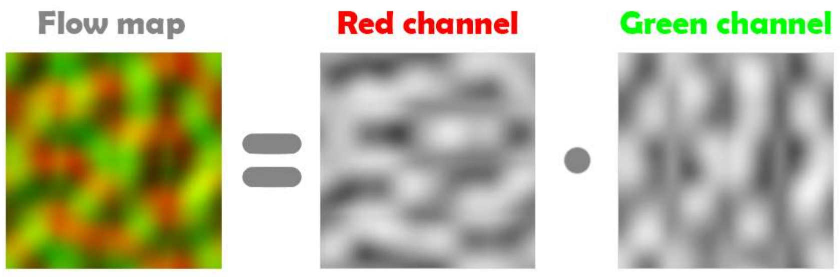

Figure 8. Flow map on the left (Flick 2017) and it’s R and G channels

The process how UVs are distorted using the color channels is quite simple but can be complicated to understand. Essentially, the black to white value that a pixel has will dictate the position where the distortion is taken from. Unmodified black and white values for U and V vectors can be seen in the example 1 found in Figure 9. Multiplying those channels will give us a color representation of the UV space. Example 2 demonstrates how a texture is tiled using this method. (Flick 2017, Moran 2016, Vleer 2017.)

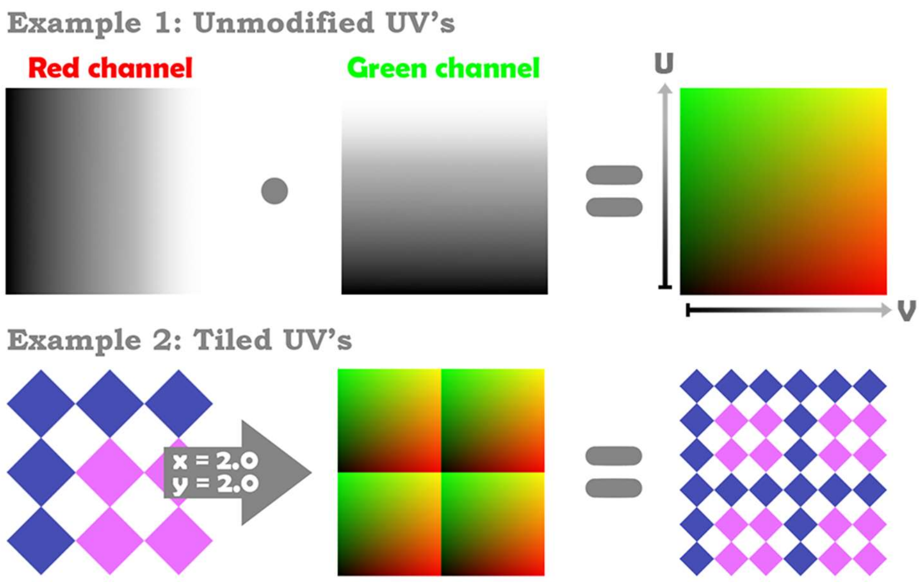

Figure 9. Examples on how a flow map works

This method can be used to distort and change the look of various effects—fire, smoke, water, etc. It can also be used on UI (User Interface) and other screen space elements to make them glow or deform. For a distortion effect to appear animated it needs to be moved in some way over the main texture (or the previous effects in the shader).

2.2.3.2 Panning and multiplying the textures

Simple movement or texture panning can be added by moving a texture over a meshes UV coordinates as seen on Figure 10. This is done by using a velocity vector to control the direction and speed of the flow. This works well for 2D and 3D objects and can be used as it is in many places where a scrolling texture is needed. The effect on its own is quite simple and often have some other effects added on top.

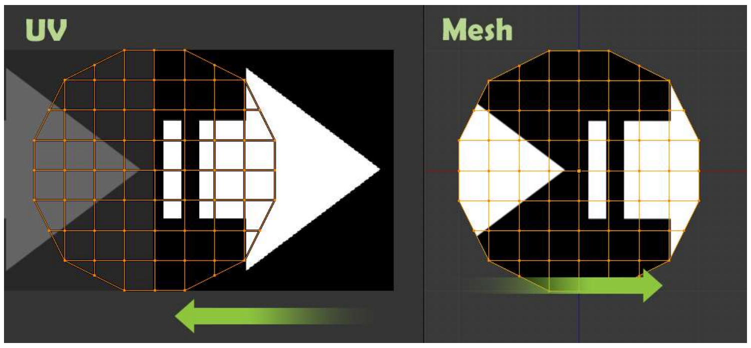

Figure 10. Moving a texture over the UV coordinates will also move it on the meshes surface

In the talk “The VFX of Diablo” Julian Love (2013) explains how they are using 3 copies of the same moving alpha (A) channel of the texture multiplied on top of each other to make an effect that seems to have irregular movement. The trick is to have one single texture scaled in varied sizes and each copy to move at different speeds (TEX1, TEX2 and TEX3 in Figure 11). Visual effects are constantly using tricks to achieve something or to decrease the cost of the effect. The second texture used in Figure 11 is TEX4 where the effect is masked out with multiplying the alpha channel with the previous steps in the shader. The final particles in the effect will then have a round and smooth edge. (Love 2013.)

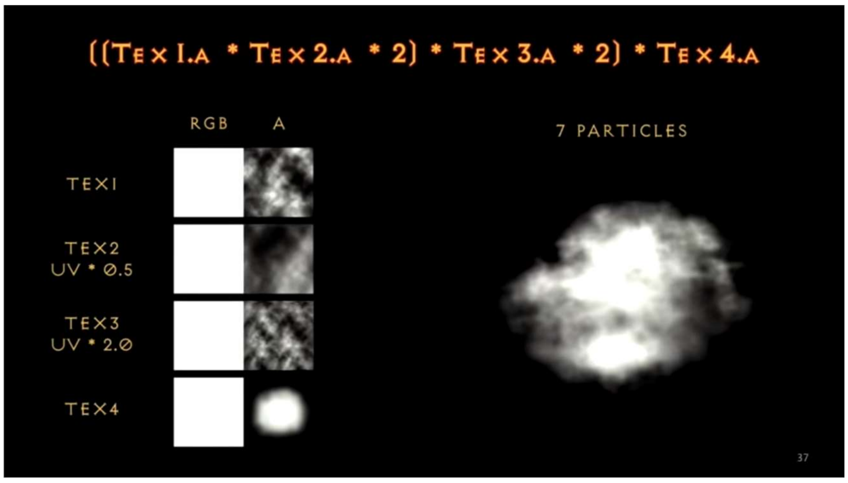

Figure 11. Moving textures multiplied on top of each other to make a smoky effect (Love 2013)

Love continues to demonstrate other effects for Diablo 3 where they are using this for different effects like smoke, fire, frost, poison, sand and so forth with different images in the textures’ RBG and alpha channels. Some effects that they use more have a bigger budget to be spent on textures, performance and the time spent on development. Some of the fire effects in Diablo have more variations in the textures, since fire is a very common effect in Diablo games. (Love 2013.)

2.2.3.3 A look into the water of RiME

Water can be a complex environmental effect when it is portrayed in realistic ways. There are effects layered on the surface of the sea: waves, foam, reflections, refractions. And then there are the effects that come from either as the causality of water movement or the interactions with water: caustics, splashes, foam particles, wetness, distortion effects, bubbles and so forth.

In Simon Trümpler's (2018) presentation "Stylized VFX in RiME", he explains the approach his colleague Pablo Fernandez at Tequila Works has taken for creating believable water for their stylized game RiME. While most of the following effects are not straight up fun feedback effects they demonstrate how effects (Figure 12) are combined together to make something that seems complicated at first. And most of these effects can be used to enhance feedback effects.

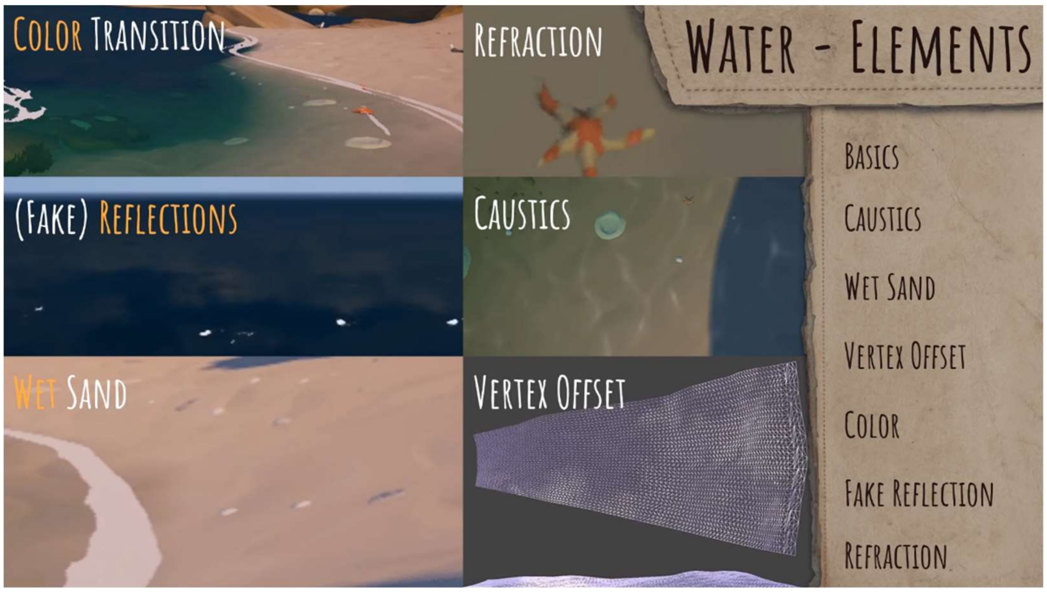

Figure 12. The effects that make the water in RiME (Tequila Works 2018)

Caustics are the reflected light rays that are made by the surface of the water. In RiME, they are shown on all of the objects that are under the water level: sand, rocks and so forth. Two notable shader effects used here are UV distortion and tri-planar mapping.

UV distortion as explained earlier makes the fake light rays move in interesting ways, which are then projected on the objects that are under the water with tri-planar mapping. This effect is blocked with a world space mask to not cast caustics above the water level. The wet sand (Figure 12) that elevates and retracts with the water is also done with a similar world space mask. Some color is multiplied on the sand and the mask is animated to go up and down in a wavelike fashion. (Trümpler 2018.)

Tri-planar mapping allows texturing without UV coordinates and it can be used to blend textures together in world space. It is also commonly used on large models like terrains that are problematic and not performant if done by unwrapping the model in UV space. (Unity 2018.) Projecting textures can also be used to enhance effects.

Offsetting vertices will add movement in the 3D mesh of the water. This is a key feature in any advanced water shader. Vertex offset can also be used in 3D UI elements or other effects like explosions, smoke and so forth. The wave movement itself is done with the help of Gerstner wave equations and a bunch of other shader tricks. Coloring the shore-line is done with the combination of tinting the transparent water, depth fade and a fresnel mask. Then some faked reflection and refraction to the water is added. (Trümpler 2018.) Plenty of shader tricks were just briefly mentioned here and some of them will be more useful than others in the world of real-time visual effects.

2.2.4 Fading in and out

Fading things in or out of the environment is a fundamental step in every visual effect. It is important to make a transition that is appropriate and feels good. Sometimes the effect suddenly pops in and out without any intro or outro, but usually the effect’s transparency or scale is changed. This is something that is often done to the particles in a particle system, but there are some ways that it’s being used for shaders and meshes.

2.2.4.1 Dissolving with a shader

Dissolving a 3D model or an image can be done by adding a greyscale map which de-termines the parts to be masked out. This is done via a threshold operation that either shows or hides a pixel based on the value between 0 and 255. The grayscale map will be used against an animated value which will make the brighter pixels opaque and ignore the darker ones. This will create a threshold-based animation as seen in Figure 13. The developers at Alkemi Games (2014) are using two different greyscale maps that are put into the R and G channels. These are then used to create a softer outline for their explosive smoke effect in the shader. (Alkemi Games 2014.)

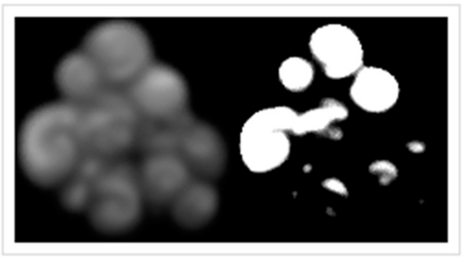

Figure 13. Black to white values tells the shader what to hide first (Alkemi Games 2014)

This technique is also often used with a tiled texture on fading characters or other 3D models. In Figure 8 there is an image of a dissolve shader's node network where the dissolving is combined with a red glowing outline of the dissolved area. This makes an interesting effect that could for example be used on burning paper that turns into black ash after it has burned until the end.

Shaders can also be used to just drive the overall transparency and scale values of the object.

2.2.4.2 Transparent vertices in a mesh

It is possible to paint the color of a vertex in a 3D modeling software. The feature itself is quite old and basically it just assigns a RBG and alpha value to a vertex (Alkemi Games 2013). Painting vertices can be used to do many things like painting a 3D model's surface, painting different textures on a single terrain, painting animation weights on a character to define how bones move the mesh or in this case to defining what the transparency value will be on each vertex.

An effect can have a mesh where the UVs are being scrolled from side to side with a shader. Vertex alpha values are then changed to fade the effect in from the side, rather than making it appear where the mesh starts as seen on Figure 14. For the effect to only run once the texture must be set to clamped in the game engine, otherwise the movement will repeat endlessly. (García-Obledo 2017.)

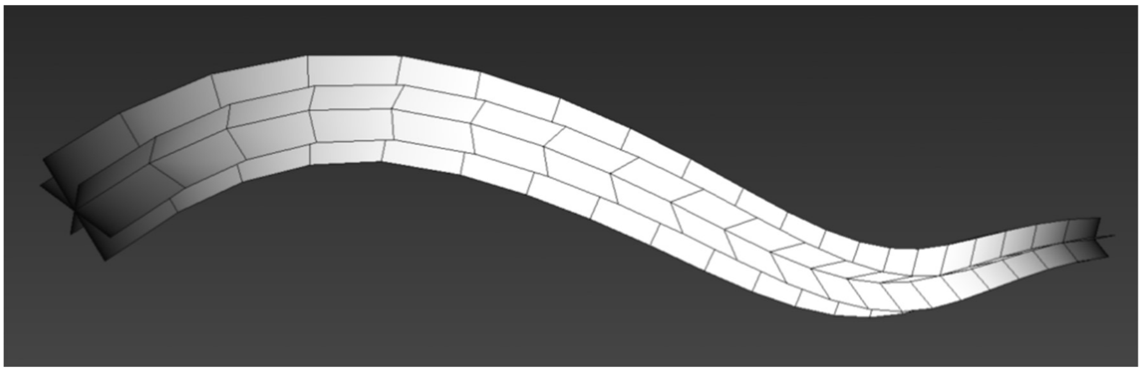

Figure 14. Making a transparent fade by coloring the vertices (García-Obledo 2017)

3.1.4.3 Opacity and size values with particle systems

Many effects use particle systems to spawn particles and a developer has the tools to change a particle’s transparency and size values during its lifetime. These will both be in constant use to introduce or remove particles from the scene. Some effects like flashes and sparks might just spawn the image right in, since the time to fade in or the total screen time might be as low as few frames.

2.2.5 Particle systems

A particle is a simple image or a mesh that is being spawned by a particle system. Usually many particles are being created to form a visual effect and they are often combined with other effects. “Each particle has a predetermined lifetime, typically of a few seconds, during which it can undergo various changes.” (Unity 2018.) Particles can have collisions that alter their direction or destroy them. Particles can also be controlled by applying forces that represent gravity, wind, magnets and so forth.

A particle system has many properties that control how the particles behave. These properties will decide how the particles are spawned and how they move after that. In Unity the properties or parameters that can be changed are shown in the editor. By editing parameters like duration, size, speed, rotation, color, dampening, etc. and how they change during their lifetime, we can achieve unique and stunning particle effects. And while Unity as well as Unreal Engine 4 offer great particle editing possibilities out of the box, the effects can be modified further by writing custom code.

2.2.6 Particle system examples

In many games, particles are being used in almost every part of the games visual look. In Assassin’s Creed Origins, particles are a big part of the environment, interactions, combat, rewarding, guiding and UI as seen on Figure 15. Visual effects, in this case particle systems, are used to better display which objects have rewards to be picked up (1), to juice interactions and movement within the world for the players to enjoy (2) and to highlight meaningful data in the user interface like experience gains, level ups and newly added quest objectives (3).

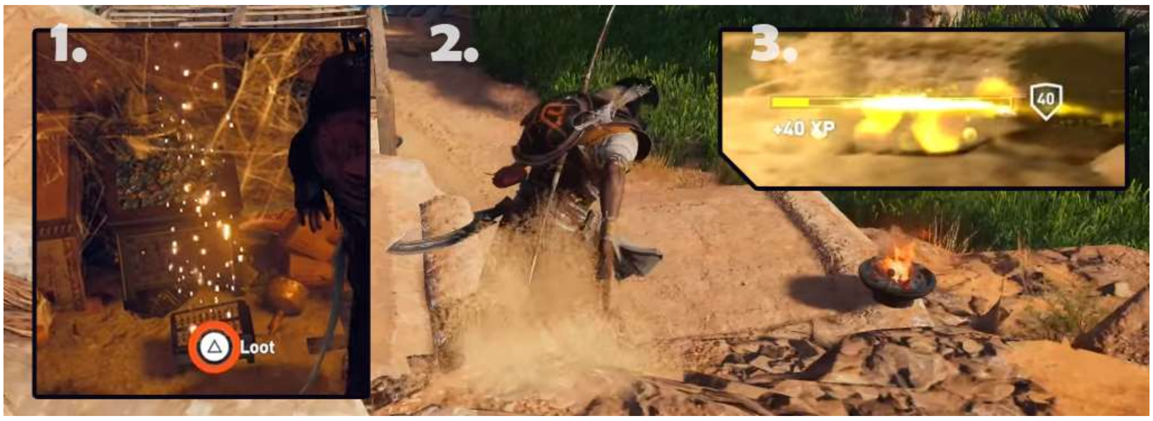

Figure 15. Screenshots from Assassin’s Creed Origins (Ubisoft Montreal 2017)

Games with higher budgets often have the graphical areas well covered and the world seems more living. However, smaller studios should also try to add movement into their games. In the pixel-world of Terraria, Re-Logic (2015) have this well covered with blinking light effects, animated enemies and the many particle systems.

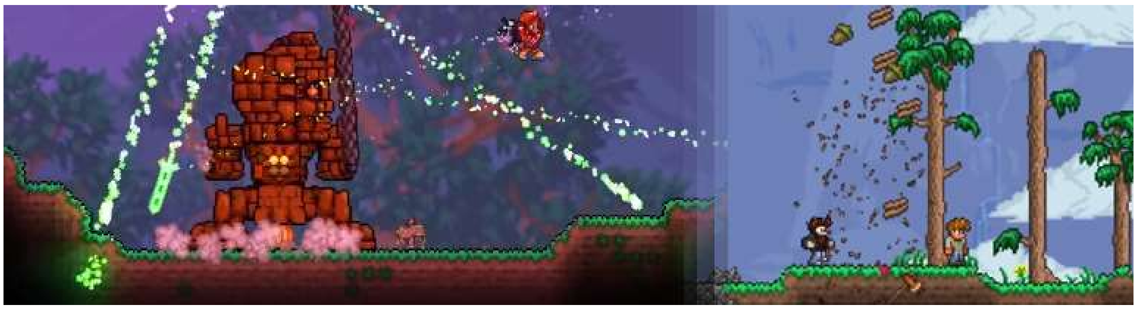

Figure 16. Pixel particle systems for a pixel world. Terraria 1.2 (Re-Logic 2015)

Effects should fit in the world and they should somehow add to the values for that specific game. More realistic effects are needed at every corner when the creators want to achieve something immersive in a realistic setting. Different stylized graphics styles can require less work and are thus better options for smaller studios. The deeper the effects the deeper the pockets need to be.

When doing particle effects, it is good to know the strengths of a particle system. They can easily add variations to our emitted particles. Having slight color differences and variation in the particles rotation, size, direction and movement can build up the visual complexity just right. Jeff Kuipers (2016) from Electronics Arts explains how the usage of modular parts can be used to build VFX elements for variety and efficiency. Using a few high-resolution textures will produce better quality and more variation to the effects when the developers need to think about performance issues. (Kuipers 2016.) Having many particle systems layered on top of each other can really help the visual fidelity as seen in Figure 17.

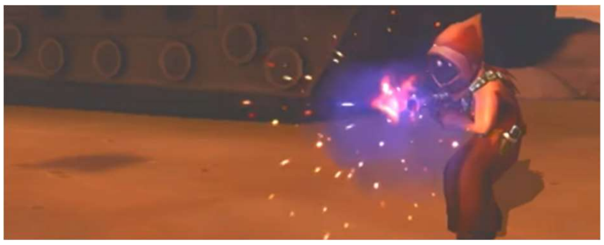

Figure 17. Multiple particle systems layered in Star Wars Galaxy of Heroes (Electronic Arts 2016)

2.3 Animations

An animation is the effect of motion accomplished by showing multiple images in a sequence. In both traditional and computer animations we have inbetweening. This process generates the missing frames between two key images or transformations the animator has set up. These keyframes are used as help to either draw the rest of the frames in traditional ways or given to a computer to calculate the missing images. (Williams 2001: 48, Vail 2017.)

In Unity, the animation curves are controlled with keys that are then used to calculate the missing frames. A keyframe is a frame that has one or more keys (Unity 2018). Animated objects can go from point A to point B in a straight line using a constant speed.

The animation can be made more interesting by varying the movement speed of the object during the animation, while keeping the length of the animation the same (Williams 2001: 49-51). Animation curves can be controlled using splines, which is an interpolation technique that calculates the frames for the animation (Komppa 2018). Splines and other interpolation techniques can be used to smooth out the movement for changes in position, size, rotation.

2.3.1 Keyframe animations

Most times simple keyframe animations are enough for juicy feedback effects. Artists can create visual effects by just animating a change in objects position, scale, color, visibility, etc. Buttons are often scaled smaller when they are pressed. A wiggly animation where the X and Y scales are alternated can be added as the release animation for a button. Parts of the user interface can be animated however wanted to appear on the screen as an action from something.

Animation can be used to activate all sorts of things like other animations, prefabs, particle systems and audio. Combining animations with other elements is useful. For example, you could have one prefab that has an introduction animation and a particle system which both activate on startup. You could then have a separate animation to activate multiple copies of this prefab. By doing this the developer only has to do the effect animation once. On a side note, spawning prefabs that have animations, particle systems, sounds, etc. is very common in game development.

2.3.2 Flipbook animations

A flipbook is “a sequence of textures compiled to one image. The pixel shader on the sprite moves the UVs to the different sections of the image and thus displaying different frames of the sequence.” (Glad 2016). This can be done with an animation, in a particle system or with code. It is usually used when the effect requires complicated changes in its form.

Flipbook animations have been used in older game titles since they are cheaper to calculate. Today computers have more processing power and can use skeletal animations for characters instead. But flipbook animations are often used for animations where the form changes and skeletal animation is to no use. Smoke and fire are good examples where a flipbook can be used.

2.4 Screenshake, delays and permanence

In the talk “The Art of Screenshake” Jan Willem Nijman (2013) presents many tricks for making a game better by showing them in a demo project. The shooter game starts with just the basics but is improved by adding features in a similar fashion as Jonasson and Purho (2012) did in their talk “Juice it or Lose it”. Effects for shooting the gun are added, bullets are made bigger and knockback from bullets is added. Enemies have a slight delay when they are hit to highlight the effect with a pause. The bodies of the enemies and the bullet shells are left on the battleground to establish a feeling of permanence. (Nijman 2013.)

Screenshake he describes as an easy way to add game feel. A screenshake is done by nudging the camera to different directions, making the screen to shake. Depending on the shake amount, direction and the length of the effect we can deliver specific feelings to the player. Shaking can be in line with the player’s shooting which makes it seem more in control. When something explodes the shaking can be all over the place. (Nijman 2013.)

2.5 Overlaid camera effects

Effects rendered on top of the camera and some post process effects can be used to give feedback as well. In Lisa Brown’s game when the players collect fireflies a screen flash is introduced. The moment the bug is captured a white overlay appears and the game is slowed down until these both gradually turn back to normal. The delay is again here to make that hard to explain feeling of anticipation and lets the player to absorb the moment. (Brown 2016.)

Chromatic aberration is a common problem that occurs when a lens distorts some of the wavelengths. Blue and red colors can be seen on the edges of the objects in a photo. (Mansurov 2011.) This effect looks interesting and it has been brought to games as a visual effect. It can be on all the time, but it is often timed as a distortion effect after something meaningful. Nicolae Berbece (2015) for example uses this in his game Move or Die, as a way to highlight player death which happens all the time (Berbece 2015). Distortion, bloom and other visual effects rendered on the main camera can be used in combination with other effects to create a fun and interesting effect for your game.

2.6 Tactile effects

“In Ico the player must guide the young girl by holding her hand. The controller vibrates creating a visceral connection between player and character.” (Lazzaro 2004)

Controller vibration is used in games that are played with a controller, which is more common for console games. Vibration events can be used to enhance the other feedback effects.

3. CASE MOOMIN: MATCH AND EXPLORE

3.1 The game

Disclaimer: Some of the things discussed in this chapter can change due to the iterative game development model. Improvements on some effects and new additional effects are planned when the game gains more success.

Moomin: Match and Explore is a mobile game from Snowfall Ltd. where players go on an adventure with the Moomin family. There are many areas to explore and an engaging story to follow. Players gather resources from levels that they complete by playing a match-3 game as the title suggests.

The match-3 mechanic used in this game is addicting and fun. And it certainly functions as an intrinsic motivator on its own. When developing a game, it is favorable to see if the underlying simulation/mechanic is satisfying and otherwise good before adding polish. There is a layer of effects added on top of the base gameplay that seeks to make it better and more fulfilling to play. It is very common for games that use some match-3 mechanic to have a lot of juice, or in other words, audiovisual rewarding (Figure 18).

Free to play mobile games usually have a multitude of different goals to push for. These are extrinsic motivators and they are often highlighted with effects to help them stand out.

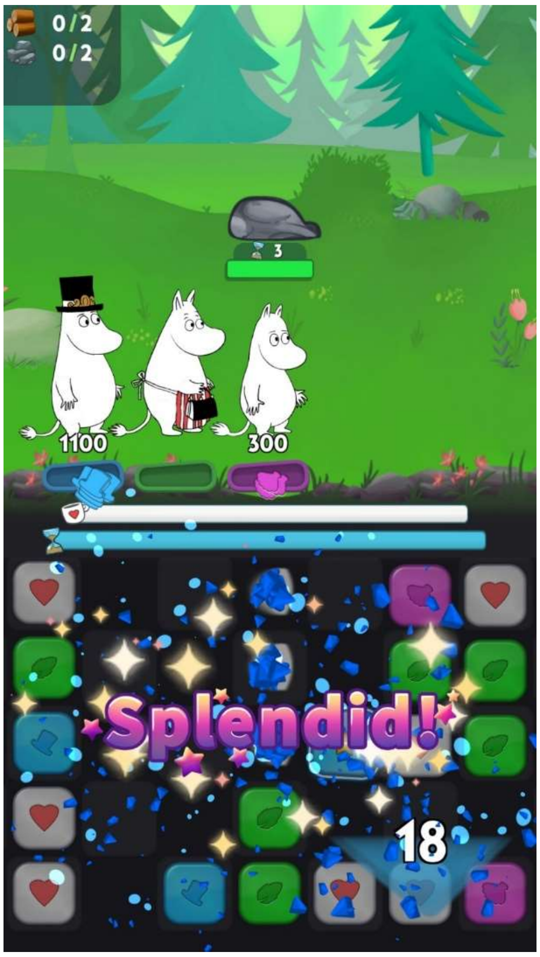

Figure 18. Screenshot from the game Moomin: Match and Explore (Snowfall 2018)

3.2 Working in a small team and good practices

Effects usually come into the picture in later stages of development and it helps to fill out another role before jumping to do visual effects. This way the project becomes familiar and the visual polish can be thought of beforehand. Previous projects that the company have done can also help with the assets. For this project there was already other graphical elements and some effects to be reused. These often need some tweaks, but they serve as a good base to start from.

Wrapping up effects that work well in the game often require knowhow and teamwork from multiple roles such as the game designer, artist, coder and sound designer. Particle systems and other effects are often spawned using code when something happens in the gameplay. Doing visual effects also require knowledge in the game engine that's being used. In this project Unity Pro was used by the developers that handled tasks inside the Unity game engine. Many of the following effects have required help from other team members. For example: all the character animations in the game are animated by another artist in the team and they work as an essential part of the experience together with many effects. Programmers also had to implement some effects at right places. But, all of the visual effects, user interfaces and their effects that we discuss from here on have been done by the author of this thesis unless otherwise stated.

When creating visual effects, it's a good idea to put them in a comparable environment. The artist can then correct possible mistakes before they go forward in the process. This was done by opening a scene with most of the gameplay, background and UI graphic assets in place and tuning the effect there. Sketching the effects beforehand helps to figure out the assets needed to complete the effect and what the process might be.

One good way is to make simple placeholder prefabs of the effects and have the coder spawn them at the correct time. Then you can create a more finalized effect and preview it in the actual gameplay. This was especially handy when creating particle effects for simultaneously breaking tiles in the match-3 gameplay.

In creative work it's important to seek out new ideas and figure out the possibilities for your effects. This includes looking into the other games on the market. This will give an idea on the expectations for your project. Additionally, some trouble spots can be avoided by looking at how other people have solved a problem.

3.3 Visual effects for the UI

The user interface in Moomin: Match and Explore uses mostly blue and golden yellow colors as seen in Figure 19. Blue acts largely as a neutral color in different backgrounds for information, text and icons. Golden yellow indicates action and reward. It’s being used in many of the buttons that lead to another UI menu and buttons that are activated or can be collected. Part of the user interface have more golden elements to indicate value. Extrinsic motivators like the amounts for stars, materials and experience collected have their own indicators to gauge player progress. Some UI elements in figure 19, have a yellow exclamation mark to notify the player that something can be collected.

Collecting something in this game makes the material or currency that is being collected to fly to their corresponding place on the UI. This type of immediate feedback can often be appealing to watch and something to await. A programmer had to implement most of this effect, since it needed to select the correct icon to display as a particle, discharge a proper amount of those particles and move them to their place in the UI. Sometimes, using code can help to control the effect and to reduce the number of premade effects.

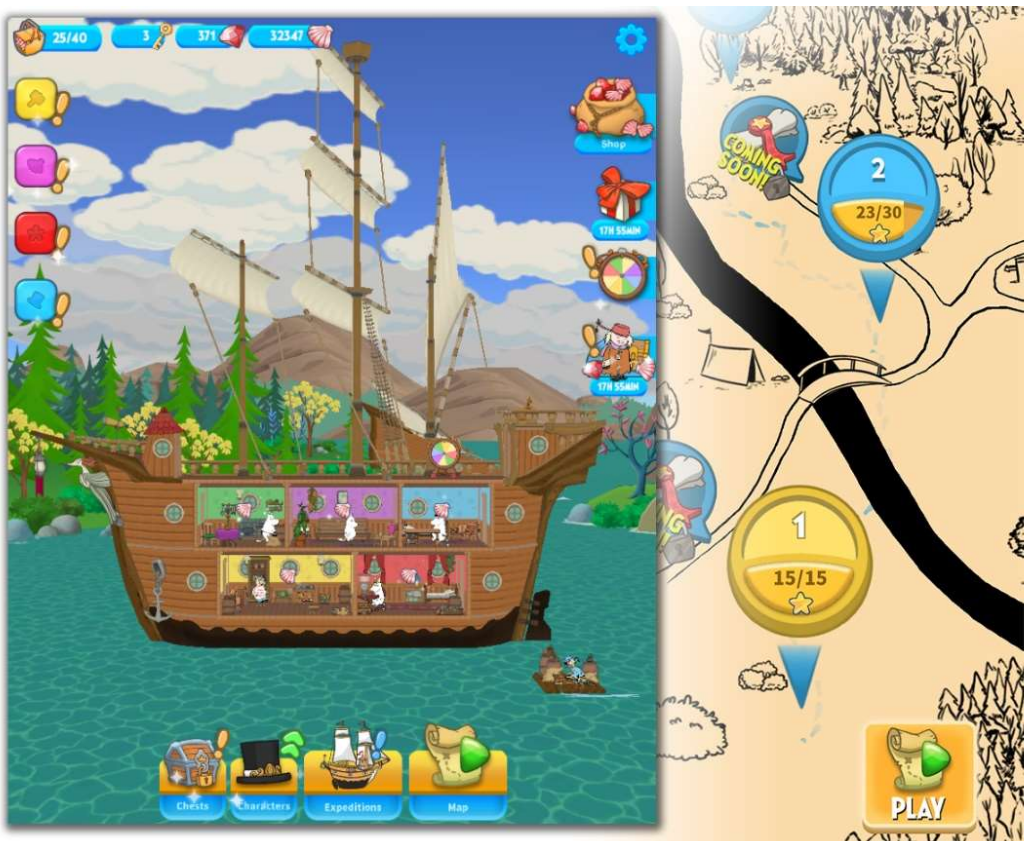

Figure 19. Match and Explore landing screen and part of the map (Snowfall 2018)

Many of the buttons in the game have some sort of animation to them. It can either be an idle animation which in many cases tries to notify of its existence or an activation animation that makes the button fun to press. One of the key elements of juice is the feedback that a player gets when they interact with the world or in this case the UI.

Some buttons as seen in Figure 20 change their color from a neutral to a more active color when toggled on. They also have a minor bouncing animation and a sound effect when pressed to make it feel good. Changing the buttons' color and playing an animation with a sound is a good example of giving juicy feedback from player interaction.

A yellow color and golden materials are something that we see often in games and they have already been associated with reward and positive things elsewhere. The green "yes" icon beneath the button is also well-known to many. By using familiar icons and the emotions behind colors, we can guide the players and improve usability. After all, recognizing patterns is our forte as mentioned previously.

Before the current UI in figure 20 came to life, the space was occupied by placeholder graphics that a programmer had put up and there were no effects at this point. A good fit was achieved when designing the UI and the effect together. User interfaces usually need more tinkering, to see which composition works best. However, getting the effect's animation to look good was a quite straightforward process, since it featured a simple keyframe animation utilizing just scaling, activating and deactivating.

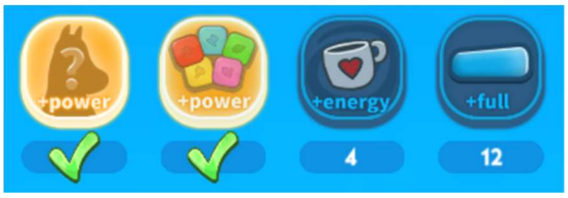

Figure 20. Buttons change color to indicate they are active or ready to be used (Snowfall 2018)

The indicator for character's special power has a pulsing highlight to inform its readiness to be used as seen in figure 21. This effect is done by adding an extra sprite on top of the slider to highlight the pulsing area. A new material for the sprite is made and its shader set to additive. An additive shader and the gradient borders in the sprite's graphics will make it glow. Pulsing is made by adding a keyframe animation to change the sprite opacity value up and down. A hat icon can also be seen in the effect. This icon varies for each color and is introduced with the rest of the effect by scaling it up from 0 to 1. These kind of visual effects, are placed to guide the player forward.

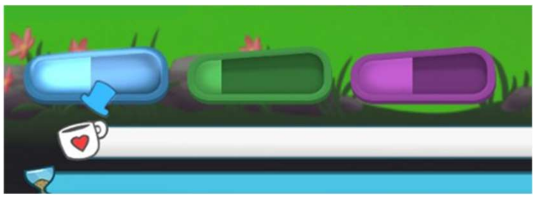

Figure 21. A simple highlight and an icon to indicate readiness (Snowfall 2018)

It’s good to have some movement when moving from a menu to another. In Moomin: Match and Explore, the camera closes in on the individual rooms when their menus are accessed. When entering the map, the game will focus on the player’s current whereabouts and opening the Wheel of Fortune menu will nudge the prize wheel slightly.

In the Wheel of Fortune menu, players can watch ads to gain spins on the wheel (Figure 22, left). When the player then interactively spins the wheel, they are rewarded with the result. This effect has a chest with glowing beams circling around it and a particle system is spawned at the start with various sparkling sprites (Figure 22, middle). The chest and its animations, together with the glowing beams were done by another artist in the team.

Similar rewarding effects are used in many places: rewards from completing a level, quick looting a level, collecting daily reward, buying a chest or using the Wheel of Fortune (Figure 22, middle and right).

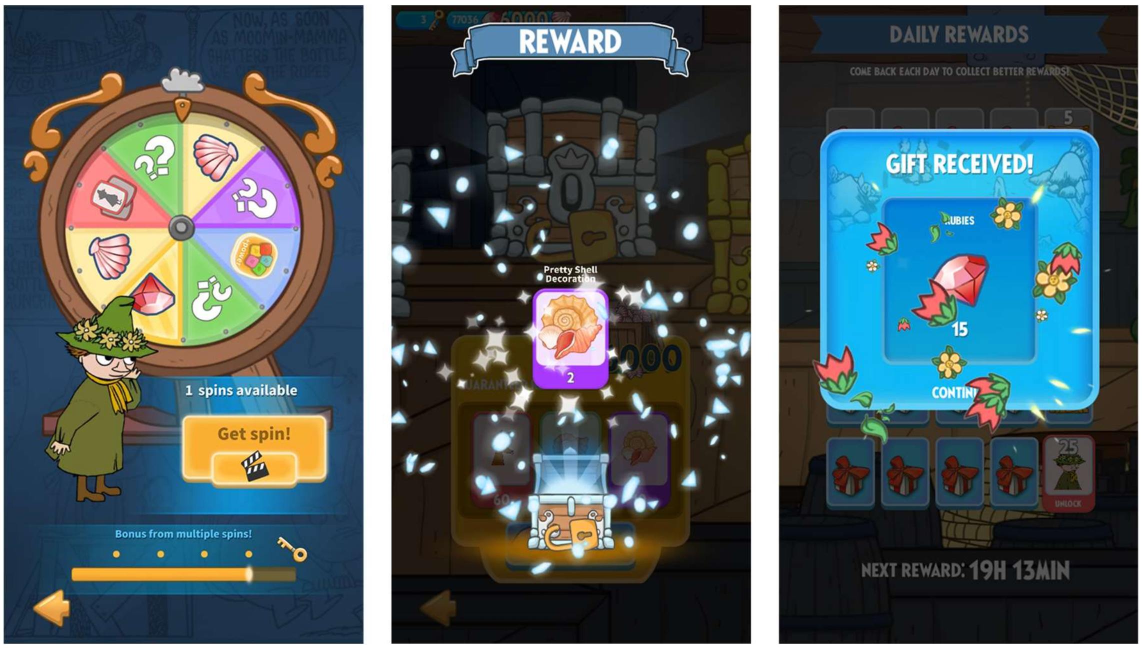

Figure 22. Left: Wheel of Fortune. Middle and right: Reward particle systems (Snowfall 2018)

After the player has completed a level successfully they are awarded one to three stars at the end of the level. This part has a longer animation where players will be rewarded. Again, there is some code running in the background, but the stars, their effects and the sparkling effect in figure 23 are from the author along with the UI graphics.

When the score slider under the stars has finished, star prefabs are activated with a slight delay between them. After this, the stars will fall on their place and each of them will shine quickly after a second or so. When the player then continues by pressing the button below, the UI will change to rewards. A chest will pop open and award the collected materials from this level. A sparking particle system will decorate the view for a brief moment, while the materials are pulled from the chest.

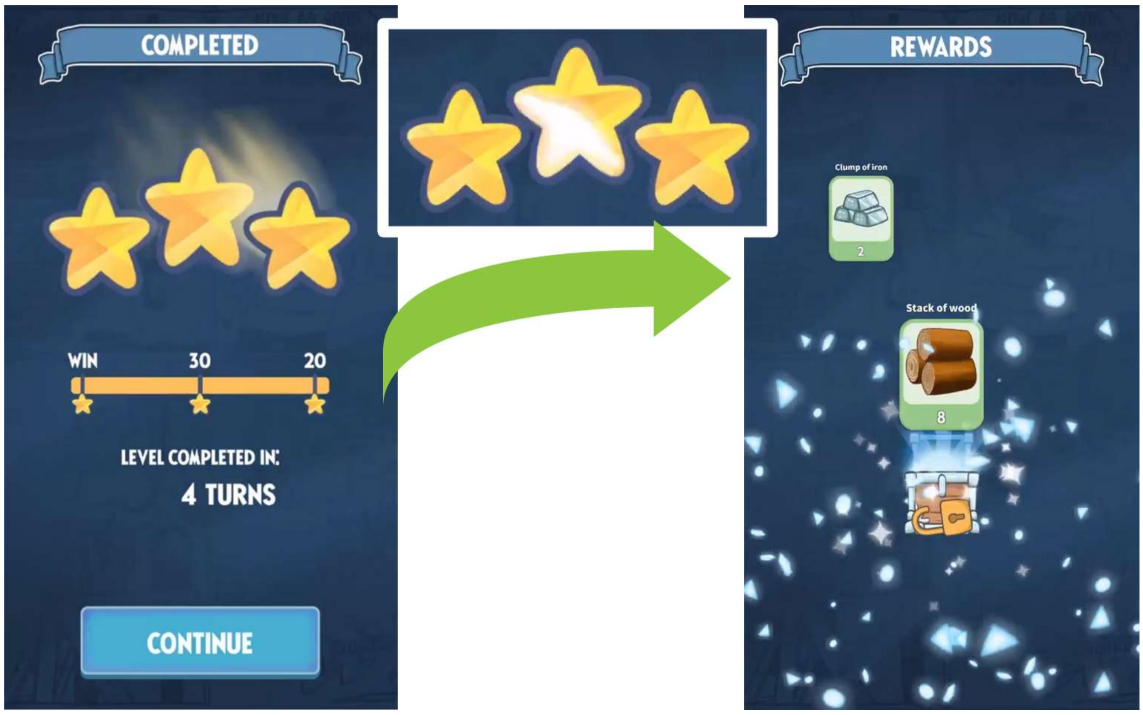

Figure 23. What happens after the level ends? (Snowfall 2018)

Each star has a keyframe animation with two sprites and the flipbook sprites for the shine (figure 24). One sprite is for the star itself and the other for the flash that can be seen on the left screenshot in figure 23. Both of these sprites will have their opacity gradually changed from 0 to 255 to introduce it like we discussed in chapter: 2.2.4 Fading in and out. They will also have some movement and rotation keyed to get them placed correctly.

When doing animations where you know the end location for the object, but the start position hasn't been decided. It is good to first key the final location, move the keyframe/keyframes until the end of the animation and then key the values that come before.

The flipbook animation controlling the shine is made in another animation. An animation controller is then used to change between the first and second part. This way we can isolate the animations and control them. In this case we only want to play the first part once and then keep replaying the second part until the player presses the button to continue.

As we can see in figure 24, the flipbook animation has five different sprites with parts of the star highlighted. These are then activated and deactivated at the right time to make the effect shine. The animation will continue to play, but it has a pause of some second until it starts again.

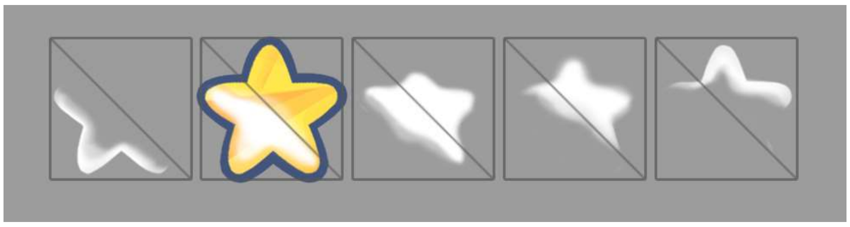

Figure 24. Images for a simple flipbook animation (Snowfall 2018)

3.4 Visual effects for the gameplay

The match-3 mechanic used in Moomin: Match and Explore has many places that allow effects to be added. Most of the gameplay takes place in the lower part of the screen, let's call it the playfield. For games in the match-3 genre, it's common to have a playfield area where tiles are moved and matched together to make matches. In Moomin: Match and Explore, there is also a top part that is used as the second part of the action in each round.

4.4.1 Top part

When players have made their matches on the lower part the characters in the top part will make their actions whether it's collecting berries, chopping down wood or something else. The actions' efficiency is based on the result from the playfield. When they hit their targets, the object will quickly flash white to give feedback from the hit. Sometimes it also spawns a particle system like dropping leaves or debris to introduce some permanence.

These targets have a health bar to indicate the progress of how far they are from being collected. A smoky puff particle system is spawned when they are collected to hide the removal of the object itself. This makes the target to drop its corresponding material from the top part to the playfield. When the material hits the playfield a smoky particle effect is spawned to highlight the new objective for the player as seen in Figure 25. This effect uses a flipbook animation of 6 frames in total to make the steps more interesting. The effect is also used on some negative gameplay elements when they are introduced to get the player's attention on them.

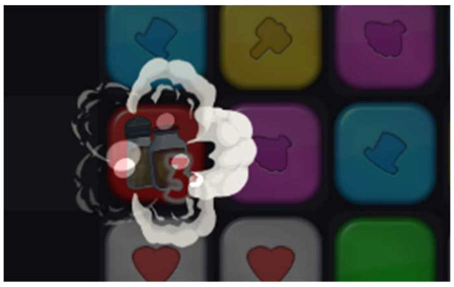

Figure 25. Particle system using a flipbook animation. 3 different frames can be seen in this picture (Snowfall 2018)

4.4.2 Playfield

Most of the gameplay consists of matching tiles with the same color together. When tiles are matched together with 3 or more, their outlines become highlighted and a particle system is activated as seen on Figure 25. Players can keep matching tiles until the timer runs out. At the end of the timer, these tiles break into pieces and are replaced with new falling tiles. On the left side of Figure 25 we can see how the matched tiles are marked for explosion and on the right side most of them have already exploded and been removed from the playfield except for the red tile that have a star icon in it. This tile is a bomb tile which will make a 3 by 3 explosion. The tiles and their glowing border were made by another artist in the team.

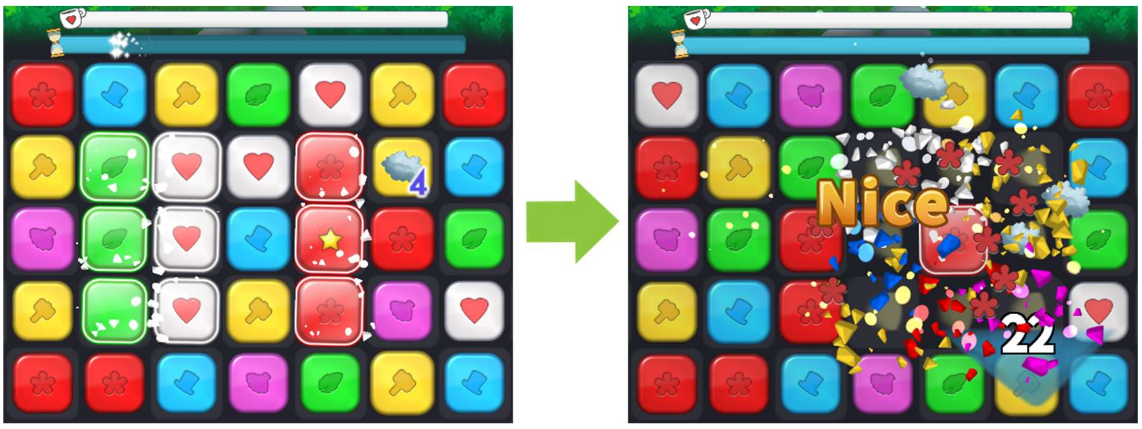

Figure 26. Tiles start to glow when they are marked to be exploded (Snowfall 2018)

These effects last on the screen for just a brief moment, but they happen all the time so it’s important to have effects that are still interesting after they have run a thousand times. When tiles break, they spawn a prefab with 3 different particle systems inside (Figure 27). The first one is a basic flash which will quickly pop in and then fade out quickly. The second one spawns several tile pieces at different sizes, rotations and velocities. They have a downward force and rotation over their lifetime to make the effect seem to have gravity. The third effect spawns some additive flakes that swivel upwards. They aim to add additional appeal and perhaps a slightly magical element to the overall effect.

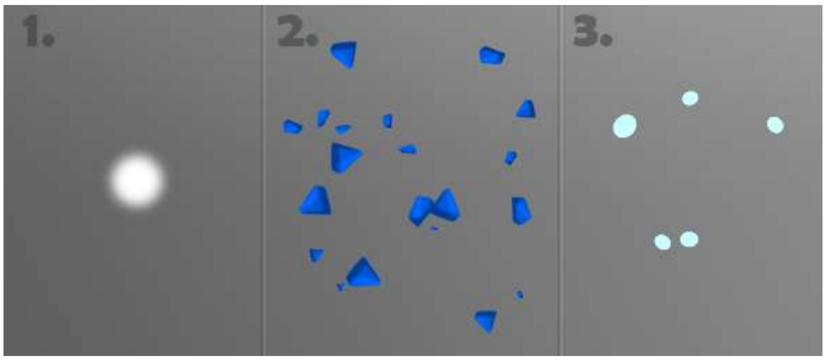

Figure 27. Components for a particle system (Snowfall 2018)

If the amount of exploding tiles is large enough the player will get an additional rewarding text saying: “Nice”, “Amazing” or “Splendid” (Figures 18, 26). This effect praises the player based on their performance. Matching tiles and looking at the effects and animations can be a fun and pleasant task for many people and hopefully an intrinsic motivator that players want to continue doing.

3.5 Visual effects for the environment

Environmental effects in games make the world seem richer and they add movement to what otherwise might seem a bit dull. There are events in the game that occasionally spawn in the world and by clicking them players will get a reward. For example, there can be events like a pirate bypassing the ship with his raft (Figure 19), a bird flying or a floating mysterious chest. Animated and interactive elements like these make the world feel alive and responding.

When the obstacles on the screen have been finished and cleared during the gameplay, there will be a sprite with treelike shapes passing the top part of the screen (Figure 28, first part). By doing this transition we can change the new objectives behind the layer or make the level end. There are also two particle systems included in this effect.

One will spawn small particles that seem to fly from the side along with the trees like some dust that is elevated by the movement. These will rotate upwards slightly while moving towards the other side and then disappear. A few can be seen in the first part of Figure 28, but at this point most of them are gone. The second particle system has leaf particles that are left from the tree transition to whirl and fall using a downwards force and random noise. These combined make the screen transition look interesting and somewhat different each time.

There are some weather effects in the environment, like: rain, wind and waves that populate parts of the scenes as seen in the second part of Figure 28. These particle systems help to create immersion and make the environment more stimulating.

Water in the game is quite simple and stylized like everything else. Its UVs are being scrolled from side to side and a flow map is added to distort the graphics slightly (Figure 28). This is done using the same methods as mentioned in the section 2.2.3 Shaders. The shader code for the water effect required help from a programmer.

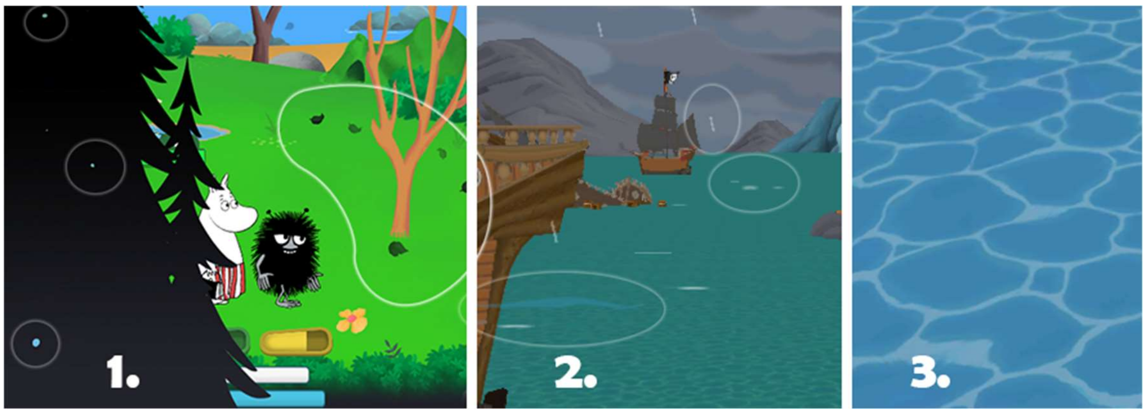

Figure 28. Environmental effects in the game (Snowfall 2018)

3.6 Author’s thoughts, conclusions and things learned

Creating visual effects and adding small details takes up some of the development time, but this last step polishes the product to look shiny and to feel good. Good game design, a well thought tutorial and a good flow between the different areas of your game is essential for a working game. But guiding the players can be aided using audiovisual effects and rewards. Giving feedback is important for keeping user interest and to express their actions. If juicy feedback effects can introduce fun elements to a game, make it more interesting and guide the players — it seems like a thing worth pursuing? Developers seem to agree that game productions should have a portion of their development time for the polish phase — which consists of juicy elements among other things...

The science behind psychology and how humans think is very fascinating, this is something that gets tangled with everything, juice too! Game artists are marketers and they are supposed to develop art and other things that the users want to experience. By studying the psychology and common practices in our field, we gain better understanding of the user’s conscious and unconscious expectations. We had a quick look into this subject in the first section of this thesis. There were good notes to take and yet, so much more to look into.

Most of the tools that were used to create the visual effects for Moomin: Match and Explore were already familiar from the start. Photoshop and Unity have been in use for many years by now, but some workflows, regarding the creation of visual effects, strengthened. One of the tools that would have been beneficial to know better is one of the visual shader node editors mentioned earlier. Unity’s own shader graph editor is quite new and not yet fully ready. But, at least Amplify Shader Editor is something that could be learned in the near future, along with the math functions and features related to shaders.

It is good to think ahead when planning visual effects. There was already experience from making VFX from earlier projects, but some new tricks and workflows were learned during this thesis. Those parts, where the UI and its effects were planned together, worked well. It also helped a lot to know the features for the UI ahead and have placeholder graphics placed. Having placeholder visual effects where they mattered, was also a good idea. Updating those prefabs and testing them in the actual gameplay proved worthy. So, having a programmer set up the placeholders is handy.

When we put more hours into working out the details of our latest visual effect or watch the effects that other people post to the web, we learn. Being an artist takes a lot of experimenting and studying, which never really ends. The visual effects for the game Moomin: Match and Explore were good and we were happy with them. But, of course there are corners that would need more polish. If the game gains more success, those corners might just get their dose of juice.

Is the game fun and do the effects make it better, fuller and more rewarding? Are there enough points of interest? These are very tricky questions. For someone the game might seem finished, but still feel lacking in some ways. When we accumulate the hours into the practice we start to see the rough corners and impurities. Minor effects, artistic intuition and the hours put into the practice can really make the difference. This also means that we should try to surpass our previous productions in the future.

Did these effects make the game better? In the end these changes were made in conjunction with many other changes and any data from the analytics wouldn’t be accurate enough. But, the team members and other users that have tested the game before and after the changes, said they liked the added visual effects.

Most of the games on the most profitable mobile games top list are heavy on effects and their quality is state of the art. One of the reasons they are so polished is because mobile free to play games are often worth developing further after their initial release if they see success.

The expectations for myself increased a lot when studying the VFX professionals and their tricks. There is always room for improvement in the world of visual effects.

REFERENCES

Koster, R. 2013. A Theory of Fun for Game Design, 2nd ed, Sebastopol, CA: O'Reilly Media Inc, pp. 40-42,73,94,124,146.

Dillon, R. 2010. On the Way to Fun: An Emotion-Based Approach to Successful Game Design, Natick, Massachusetts: A K Peters/CRC Press, p.15.

Swink, S. 2009. Game feel: a game designer's guide to virtual sensation, Burlington, Massachusetts: Morgan Kaufmann Publishers, pp. 5,151.

Zoss, M. 2009. The Art Of Game Polish: Developers Speak. Referenced on 07.09.2018. https://www.gamasutra.com/view/feature/132611/the_art_of_game_polish_developers_.php

Gray, K., Gabler, K., Shodhan, S., Kucic, M. 2005. How to Prototype a Game in Under 7 Days. Referenced on 06.09.2018. https://www.gamasutra.com/view/feature/130848/how_to_prototype_a_game_in_under_7_.php

Forestié, R., Gameloft Montreal. 2018. Best Practices for fast game design in Unity – Unite LA. Referenced on 05.12.2018. https://youtu.be/NU29QKag8a0

Jonasson, M., Purho, P. 2012. Juice it or lose it. Referenced on 14.09.2018. https://www.youtube.com/watch?v=Fy0aCDmgnxg

Brown, L. 2016. Vector 2016 – The Nuance of Juice Talk. Referenced on 07.09.2018. https://www.youtube.com/watch?v=qtgWBUIOjK4

Lazzaro, N. 2004. Why We Play Games: Four Keys to More Emotion in Player Experiences. Referenced on 12.09.2018. http://twvideo01.ubm-us.net/o1/vault/gdc04/slides/why_we_play_games.pdf

Tyson, N. d. 2015, Cosmos: A Spacetime Odyssey (Episode 3), When Knowledge Conquered Fear.

Joosten, E., van Lankveld, G., Spronck, P. 2010. Colors and Emotions in Videogames. Referenced on 13.09.2018. https://www.researchgate.net/publication/239842533_Colors_and_Emotions_in_Video_Games

Cherry, K. 2018. Color Psychology: Does It Affect How You Feel? Referenced on 13.09.2018. https://www.verywellmind.com/color-psychology-2795824

Nintendo. 1998. The Legend of Zelda: Ocarina of Time

Color Matters. The Meanings of Blue. Referenced on 13.09.2018. https://www.colormatters.com/the-meanings-of-colors/blue

Cook, J. 2015. The Color of Enlightenment: Defining Shangri-La Within Far Cry 4. Referenced on 13.09.2018. https://www.youtube.com/watch?v=XUz7ZiR0RvY

Turner, J. 2015. Oh My! That Sound Made the Game Feel Better! Referenced on 13.09.2018. https://www.gdcvault.com/play/1022808/Oh-My-That-Sound-Made

Ryan, R. M., Deci, E. L. 2000. Contemporary Educational Psychology, Intrinsic and Extrinsic Motivations: Classic Definitions and New Directions. Referenced on 26.09.2018. https://www.sciencedirect.com/science/article/pii/S0361476X99910202

Nielsen, J. 1995. 10 Usability Heuristics for User Interface Design. Referenced on 31.10.2018. https://www.nngroup.com/articles/ten-usability-heuristics/

Zichermann, G. 2011. Intrinsic and Extrinsic Motivation in Gamification. Referenced on 26.09.2018. http://www.gamification.co/2011/10/27/intrinsic-and-extrinsic-motivation-in-gamifica-tion/

Guerrette, K. 2018. Real-Time VFX: Overview from Keith Guerrette. Referenced on 27.09.2018. https://80.lv/articles/real-time-vfx-overview-from-keith-guerrette/

Kuipers, J. 2016. Unite Europe 2016 – Jedi Tricks for Visual Quality on Mobile. Referenced on 27.09.2018. https://www.youtube.com/watch?v=YxrF4R_p4GM

Ordóñez, F. 2017. VFX for Games Explained. Referenced on 31.10.2018. https://80.lv/arti-cles/vfx-for-games-explained/

Keyser, J., Riot Games. 2018. Artistic Principles of VFX #5 Timing. Referenced on 05.12.2018. https://www.youtube.com/watch?v=WLMVpcK0WvA

Jevremović, S. 2018. VFX Staples: Shape, Color and Motion. Referenced on 05.12.2018. https://80.lv/articles/vfx-staples-shape-color-and-motion/

de Laat, Sjors. 2018. Learning VFX Workflow. Referenced on 05.12.2018. https://80.lv/arti-cles/learning-fx-workflow/

Riot Games. 2017. The Complete Guide to Creating Visual Effects within League of Legends. Referenced on 27.09.2018. https://nexus.leagueoflegends.com/wp-content/up-loads/2017/10/VFX_Styleguide_final_public_hidpjqwx7lqyx0pjj3ss.pdf

Unity Technologies. 2018. Unity Documentation - Texture Types. Referenced on 28.09.2018. https://docs.unity3d.com/Manual/TextureTypes.html

IT Hare. 2016. Game Graphics 101: Textures, UV Mapping, and Texture Filtering. Referenced on 01.10.2018. http://ithare.com/game-graphics-101-textures-uv-mapping-and-texture-filtering/

Marmoset. 2015. Physically-Based Rendering, and You Can Too! Referenced on 01.10.2018. https://marmoset.co/posts/physically-based-rendering-and-you-can-too/

Polycount wiki. 2018. ChannelPacking. Referenced on 10.12.2018. http://wiki.poly-count.com/wiki/ChannelPacking

NVIDIA Corporation. 2018. Pixel shaders. Referenced on 10.12.2018 https://www.nvidia.com/ob-ject/feature_pixelshader.html

Khronos Group. 2015. Shader. Referenced on 10.12.2018. https://www.khronos.org/opengl/wiki/Shader

Unity Technologies. 2018. Unity. Referenced on 10.12.2018. https://unity3d.com/unity

Epic Games. 2018. What is Unreal Engine 4. Referenced on 10.12.2018. https://www.unrealen-gine.com/en-US/what-is-unreal-engine-4

Okulov, V. 2018. Unity and Unreal Engine Comparison pt.3. Referenced on 10.12. 2018 https://not-lonely.com/blog/making-of/unity-vs-ue-pt-3/

Unity Technologies. 2018. Unity Documentation - Creating and Using Materials. Referenced on 01.10.2018. https://docs.unity3d.com/Manual/TextureTypes.html

Epic Games. 2018. Unreal Engine 4 Documentation – Essential Material Concepts. Referenced on 01.10.2018. https://docs.unrealengine.com/en-US/Engine/Rendering/Materials/Introduction-ToMaterials

Autodesk. 2018. UVs. Referenced on 10.12.2018. https://knowledge.autodesk.com/support/maya/learn-explore/caas/CloudHelp/cloudhelp/2018/ENU/Maya-Modeling/files/GUID-FDCD0C68-2496-4405-A785-3AA93E9A3B25-htm.html

Bech-Yagher, C. 2017. UV mapping for beginners. Referenced on 05.10.2018. https://www.creativebloq.com/features/uv-mapping-for-beginners

Unity Technologies. 2018. Shader Graph Example Library. Referenced on 02.10.2018. https://github.com/UnityTechnologies/ShaderGraph_ExampleLibrary

Flick, J. 2017. Catlike Coding - Texture Distortion - Faking Liquid. Referenced on 04.10.2018. https://catlikecoding.com/unity/tutorials/flow/texture-distortion/

Moran, D. 2016. Shaders 101 – Intro to Shaders. Referenced on 10.12.2018. https://www.youtube.com/watch?v=T-HXmQAMhG0

Vleer, Y. [UE4] Uv's (somewhat) Explained. Referenced on 10.12.2018. https://youtu.be/sR0xV0TDvS8

Love, J., Activision Blizzard. 2013. Technical Artist Bootcamp: The VFX of Diablo. Referenced on 03.10.2018. https://www.youtube.com/watch?v=YPy2hytwDLM

Trümpler, S., Fernandez, P., Tequila Works. 2018. 'Stylized VFX in RiME' by Simon Trümpler | Unreal Fest Europe 2018 | Unreal Engine. Referenced on 03.10.2018. https://www.youtube.com/watch?v=ExD_p3hsV80

Unity Technologies. 2018. Triplanar Node. Referenced on 08.10.2018. https://github.com/Unity-Technologies/ScriptableRenderPipeline/wiki/Triplanar-Node

García-Obledo, F. 2017. VFX for Games Explained. Referenced on 11.10.2018. https://80.lv/articles/vfx-for-games-explained/

Alkemi Games. 2014. A game of Tricks III – Particles fun (part1). Referenced on 11.10.2018. http://www.alkemi-games.com/a-game-of-tricks-iii-particles-fun-part1/

Alkemi Games. 2013. A game of Tricks II – Vertex Color. Referenced on 11.10.2018. http://www.alkemi-games.com/a-game-of-tricks-ii-vertex-color/

Williams, R. 2001. Animator's Survival Kit: A Manual of methods, principles and formulas for classical, computer, games, stop motion and internet animators, London, UK: Faber & Faber, pp. 48-51.

Vail, N. 2017. Understanding Linear Interpolation in UI Animation. Referenced on 12.12.2018. https://medium.freecodecamp.org/understanding-linear-interpolation-in-ui-animations-74701eb9957c

Unity Technologies. 2018. Unity Documentation – Using Animation Curves. Referenced on 25.10.2018. https://docs.unity3d.com/Manual/animeditor-AnimationCurves.html

Komppa, J. 2015. Interpolation tricks. Referenced on 12.11.2018. http://sol.gfxile.net/interpolation/#c8

Nijman, J. W., Vlambeer. 2013. The Art of Screenshake. Referenced on 13.11.2018. https://www.youtube.com/watch?v=AJdEqssNZ-U

Glad, A. Realtime VFX Dictionary Project – Real-Time VFX. Referenced on 01.11.2018. https://realtimevfx.com/t/realtime-vfx-dictionary-project/570

Mansurov, N. 2011. What is Chromatic Aberration. Referenced on 13.11.2018. https://photog-raphylife.com/what-is-chromatic-aberration

Berbece, N. 2015. Game Feel: Why your Death Animation Sucks. Referenced on 13.11.2018. https://www.youtube.com/watch?v=pmSAG51BybY

Figure References

Figure 2: Forestié, R., Gameloft Montreal. 2018. With or without juice. Referenced on 05.12.2018. https://youtu.be/NU29QKag8a0

Figure 4: Ubisoft. 2014. Screenshots from the game Far Cry 4.

Figure 5: Riot Games, 2017. League of Legends VFX Guide. Referenced on 26.11.2018. https://nexus.leagueoflegends.com/wp-content/uploads/2017/10/VFX_Styleguide_final_public_hidpjqwx7lqyx0pjj3ss.pdf

Figure 7: Unity Technologies. 2018. A screenshot of dissolve shader's nodes from the Shader Graph Example Library. Referenced on 26.11.2018. https://github.com/UnityTechnologies/ShaderGraph_ExampleLibrary

Figure 8: Flick, J. 2017. Flow map. Refenced on 26.11.2018. https://catlikecoding.com/unity/tutorials/flow/texture-distortion/

Figure 11: Love, J., Activision Blizzard. 2013. Moving textures multiplied on top of each other to make a smoky effect. Refenced on 26.11.2018. https://www.youtube.com/watch?v=YPy2hytwDLM

Figure 12: Tequila Works. 2018.The effects that make the water in RiME. Referenced on 26.11.2018. https://www.youtube.com/watch?v=ExD_p3hsV80

Figure 13: Alkemi Games. 2014. Black to white values tells the shader what to hide first http://www.alkemi-games.com/a-game-of-tricks-iii-particles-fun-part1/

Figure 14: García-Obledo, F. 2017. Making a transparent fade by coloring the vertices https://80.lv/articles/vfx-for-games-explained/

Figure 15: Ubisoft Montreal. 2017. Screenshots from Assassin's Creed Origins. Refenced on 26.11.2018.

Figure 16: Re-Logic. 2015. Pixel particle systems for a pixel world. Terraria 1.2. Refenced on 26.11.2018.

Figure 17: Kuipers, J., Electronics Arts. 2016. Multiple particle systems layered in Star Wars Galaxy of Heroes. Refenced on 26.11.2018. https://www.youtube.com/watch?v=YxrF4R_p4GM

Figure 18 Snowfall. 2018. Screenshot from the game Moomin: Match and Explore. Referenced on 26.11.2018.

Figure 19. Snowfall. 2018. Match and Explore landing screen and part of the map. Referenced on 26.11.2018.

Figure 20: Snowfall. 2018. Buttons change color to indicate they are active or ready to be used. Referenced on 26.11.2018.

Figure 21: Snowfall. 2018. A simple highlight and an icon to indicate readiness. Referenced on 26.11.2018.

Figure 22: Snowfall. 2018. Left: Wheel of Fortune. Middle and right: Reward particle systems Referenced on 26.11.2018.

Figure 23. Snowfall. 2018. What happens after the level ends? Referenced on 12.12.2018.

Figure 24: Snowfall. 2018. Images for a simple flipbook animation. Referenced on 26.11.2018.

Figure 25: Snowfall. 2018. Particle system using a flipbook animation. 3 different frames can be seen in this picture. Referenced on 26.11.2018.

Figure 26: Snowfall. 2018. Tiles start to glow when they are marked to be exploded. Referenced on 26.11.2018.

Figure 27: Snowfall. 2018. Components for a particle system. Referenced on 26.11.2018.

Figure 28: Snowfall. 2018. Environmental effects in the game. Referenced on 26.11.2018.