Learning Plans & Research Journal

Assignment 4 – Final Project Documentation

INFT575 Preparation of Final Business Case

Chris Murphy (CIT182880)

25 June 2019

Table of Contents

Learning Plans

Plan A

Goal: learn more about shaders and other graphical rendering techniques

Steps:

1. *Planning phase* – create a list of the retro-style rendering effects to be included as part of the project
2. *Planning phase* – research the best practices for building shaders in Unity
3. *Development phase* – research and either implement or imitate each of these effects in Unity shader form

Plan B

Goal: learn how to develop a Unity toolset and release it onto the Unity Asset Store

Steps:

1. *Planning phase* – research similar products that exist on the Asset Store
2. *Planning phase* – read the submission process for the Asset Store
3. *Development phase* – develop the product to meet the standards required for release onto the Asset Store
4. *Development phase* – release the product onto the Asset Store

Plan C

Goal: learn how to plan and complete a year-long development project

Steps:

1. *Planning phase* – complete all required research and planning documentation, with special emphasis on the project timeline to ensure that the project is most likely to stay on track
2. *Development phase* – develop the product, ensuring that due dates for work tasks don't slip and cutting features if necessary to ensure the project is finished on time

Journal Entries

Settled on project idea & created a basic retro 3D Unity shader – 12/03/19

Learning plan/s involved: A

Obviously, the first important step for this project is figuring out which of my product ideas to go ahead with. Luckily my recent work on personal projects has led me to an idea that I'm excited to pursue – I'm thinking of making some kind of shader pack or collection which provides the user with the tools to build a retro-3D aesthetic that imitates the visual style of early 3D horror games such as *Silent Hill* or *Resident Evil.* I've recently played several indie horror games that use this visual style to help evoke an indistinct, unnerving atmosphere and they inspired me to try to create the same effect as I've always enjoyed messing around with programming shaders and other visual FX.

One of these games that I downloaded and played recently is called *Paratopic,* so I poked around the included text file containing the licenses of the various tools that were used to build the game to see if I could find something useful. One of the referenced tools was dsoft20's *psx_retroshader* pack, a collection of Unity shaders which emulate the rendering style of the original PlayStation and which is freely available on GitHub. While looking through and researching the code, I also came across a similar product in Kejiro Takahashi's *Retro3D*. After combing through the lines of both shader packs I was able to comprehend and build my own basic unlit Unity shader which implements two effects in order to simulate the visual style of early 3D games:

- **vertex 'snapping' or 'jitter'** – this is accomplished by quantizing the world-space position of each vertex to imitate an effect which would occur in older 3D rendered models due to the available precision of the position value for each vertex being much lower than nowadays (Stack Exchange, 2017).
- **affine texture mapping** – an early form of texture mapping which doesn't include any perspective correction, so that as the camera moves closer to an angled surface textures will begin to visibly warp. This method of texture mapping was used on the original PlayStation before hardware became powerful enough to use techniques that included perspective correction as part of the process.

This basic retro 3D shader will serve as a basis for building more complex shaders in the future – I'll probably try to create a shader that supports Unity's inbuilt lighting system next.

Research into visual styles utilised by target developers for product – 02/04/19

Learning plan/s involved: A

Since the target audience for my product is indie game developers (especially those with a focus on making horror games), I thought that it would be a good idea to look through and analyse their output to figure out how to make my product more appealing.

As mentioned in the previous journal entry, I recently played Arbitrary Metrics' *Paratopic*, which is a short first-person horror narrative game intended to be completed in a single sitting. It uses various retro-3D visual effects such as lowered resolution, vertex jitter, unfiltered textures, and posterization/colour banding - these were implemented using dsoft20's *psx_retroshader* pack for Unity. The game is a perfect example of the kind of developer that my product will be targeted at – a three-person team building a game in Unity who would be willing to invest a relatively small amount in order to achieve their targeted retro-3D visual style.

Another indie game which uses this visual style is Sorath's *Devil Daggers*, a fast run-based arena FPS with a horror aesthetic. In addition to using several of the retro-3D effects listed above, this game uses vertex-based coloured lighting. This is an older technique used for rendering lights in scenes that was replaced by per-pixel lighting when more powerful hardware became available (*Per-pixel lighting* 2017). *Devil Daggers* was built using a custom engine rather than Unity, so this isn't an example of a developer who is likely to have used my product if it was available – however, the implementation of some kind of coloured vertex lighting in my package seems like a good idea for a potential feature.

A developer who has put out a lot of work using a retro-3D style is Puppet Combo, who develops horror games such as *Nun Massacre, Babysitter Bloodbath* and *The Night Ripper* that are inspired by old slasher movies *(Puppet Combo* (n.d.)). Alongside more straightforward retro-3D visual effects, the aesthetics of these games draw heavily from 80's VHS movies and as such include effects to make them look like movies being played on an old, damaged VHS tape such as scanlines, chromatic aberration, screen jumping, and more. These effects could possibly included in my product if I end up completing the planned effects early, or perhaps as an addon in the future.

After looking through the works of several indie developers including the ones listed above, I noticed an important common element – none of these games implement all of the effects required to accurately imitate the visual style of an early 3D game. Instead, they pick and choose several different effects and combine them with modern visual effects to create a kind of distinctive 'faux-retro' look which creates the impression of older games without unnecessarily constraining the developer. As such, rather than building my pack to implement every retro effect in a single shader, I think that it'll be far more effective and useful to make each individual effect modular and scalable. This way, developers can implement and modify only the effects that they want in order to build their own 'faux-retro' visual style.

Research into similar existing products – 15/04/19

Learning plan/s involved: B

Another important step in the process of building the initial concept for my product is to research similar existing products, in this case mostly focusing on those that can be found on the Unity Asset Store. This will help me to decide on a number of details for my product such as:

- which shaders/effects should be the highest priority to include
- how many effects should ideally be included
- the most appropriate price point – it must be high enough to offset the amount of development work that I put in, while also being low enough to be competitive on the store front
- additional ideas for shaders/effects that I haven't yet thought of

The product which is both the most substantial and the closest in concept to my idea is *PSXEffects* by Christian Kosman (Kosman 2019). It is a comprehensive package which uses around 30 adjustable effects to implement the visual style of the original PlayStation, with a listed price of $20AUD. The number of effects included in this package is roughly similar to the amount that I'm planning to include in mine, so my pricing point should probably be somewhat similar as well. It's currently sitting at a 5/5 rating on the store page – an important takeaway from the commented reviews is that while many of them praise the product itself, most of the text is dedicated to highlighting the fantastic level of customer support provided by the developer. As one user states, “I had a small issue using PSXEffects at one point and contacted the developer for support. They responded within 30 minutes and helped me fix an obscure problem within a day!” (Kosman 2019). Several of the other 5/5 comments include passages such as this – the takeaway is that providing quality post-release customer support should be an emphasis of my project in order to improve the customer ratings. Another interesting feature of the *PSXEffects* store page is that it links to a free web-based Unity demo scene which allows the user to fly around and see an in-game demonstration of each effect in action, as well as being able to pull up a menu of sliders and modify the effects themselves. This is a fantastic idea, and I'm definitely going to do the same with my own webpage if I have the time near the end of development.

Another similar product is *Retro Look Pro* by Limitless Unity Development. Rather than emulating the 3D visual style of old gaming hardware, the focus of this package is on simulating the displays of different mediums through post-processing such as CRT monitors, old TVs and VHS tapes. These are some additional effect types that I might include in my package if I have spare development time near the end, but the most important takeaway from this product is the fact that it uses a custom Unity editor inspector to allow the user to easily toggle and manipulate each of the effect values. Since one of the pillars of my product design is that it should be able to be used by non-programmers, this is a feature that will be a good idea to include. Additionally, *Retro Look Pro* is priced at $25AUD which further demonstrates a price of around $20 for my package would be both reasonable and competitive.

Most other similar products that I could find on the Unity Asset Store were much smaller, usually only consisting of one or two effects and also requiring programming knowledge to implement properly.

Research into costs – 12/06/19

Learning plan/s involved: C

Another important factor in the development of my product is figuring out roughly how much development will cost. Making sure that the total costs remain relatively low is important due to the nature of the project – the product is targeted at a niche market, so it's very difficult to estimate the likely profits with any degree of accuracy.

The most obvious cost will be in terms of time. According to the rough time estimations for each task in the Work Breakdown Structure included as part of this documentation, the time put into this project should be somewhere around 128 hours total. Currently the legal minimum wage in Australia is around $19 per hour, so the minimum possible amount that my development time on the project is worth would be around $2,432. However, according to PayScale.com the average salary for a junior software engineer in Australia is $23.85 per hour (*Average Junior Software Engineer Salary* (n.d.)), which would mean that a more appropriate valuation of my development time is somewhere around $3,052.80.

The main tools that I intend to use for development are Unity Personal Edition and Visual Studio Community 2017. According to the Unity terms of service, assets developed using Unity can be sold as long as the developer hasn't earned/received more than $100,000 in revenue or funding in the last fiscal year (*Can I Make A Commercial Game With Unity Free/Personal Edition?* 2015), so that shouldn't be a problem. Additionally, the store page for Visual Studio Community states that “any individual developer can use Visual Studio Community to create their own free or paid apps” (Microsoft 2019).

One potential cost for the project is buying similar products in order to analyse how they work and get a better idea of how best to implement my product. At this point, the two most likely candidates (as mentioned in the above 'research into similar products' entry) are *PSXEffects* and *Retro Look Pro*, the purchases of which would total $45AUD.

The final avenue for potential costs is buying assets such as models and textures for the example scene. However, as a solo game developer I'm used to finding and using royalty-free assets with the appropriate crediting as well as developing simple textures and models myself – I'll do the same in this situation in order to help save on costs.

All up, a rough estimation of the costs of the project totals around $2,500 at a minimum. According to the above research entry on similar products, the appropriate price range for my product will likely need to be in the $20-$25 range to be competitive. This means that in order to break even my product will need to sell somewhere between 100-125 licenses.

References

- Arbitrary Metric, 2018, *Paratopic*, PC game, (n.p.)
- *Average Junior Software Engineer Salary*, (n.d.), viewed 12/06/19, *https://www.payscale.com/research/AU/Job=Junior_Software_Engineer/Salary*
- *Can I Make A Commercial Game With Unity Free/Personal Edition?*, 2015, viewed 12/06/19, *https://support.unity3d.com/hc/en-us/articles/205253119-Can-I-make-a-commercial-game-with-Unity-Free-Personal-Edition-*
- Capcom, 1996, *Resident Evil*, PlayStation game, Capcom, Osaka
- dsoft20, 2016, *psx_retroshader*, computer software, https://github.com/dsoft20/psx_retroshader
- Kosman, C, 2019, *PSXEffects*, Unity Asset Store, viewed 15/04/19, https://assetstore.unity.com/packages/vfx/shaders/psxeffects-132368
- LEAKYFINGERS, 2019, *Concrete Crown*, PC game, (n.p.)
- Limitless Unity Development, 2019, *Retro Look Pro*, viewed 15/04/19, https://assetstore.unity.com/packages/vfx/shaders/fullscreen-camera-effects/retro-look-pro-133193
- Microsoft, 2019, *Visual Studio Community*, viewed 12/06/19, https://visualstudio.microsoft.com/vs/community/
- *Per-pixel lighting*, 2017, viewed 02/04/19, *https://en.wikipedia.org/wiki/Per-pixel_lighting*
- *Puppet Combo*, (n.d.), viewed 02/04/19, *https://puppetcombo.com/#games*
- Sorath, 2016, *Devil Daggers*, PC game, (n.p.)
- Stack Exchange, 2017, *Why did the old 3D games have 'jittery' graphics?*, viewed 12/03/19, https://gamedev.stackexchange.com/questions/26823/why-did-the-old-3d-games-have-jittery-graphics
- Takahashi, K, 2015, *Retro3D*, computer software, https://github.com/keijiro/Retro3D
- Team Silent, 1999, *Silent Hill*, PlayStation game, Konami, Tokyo
- *Texture Mapping*, 2019, viewed 02/04/19, https://en.wikipedia.org/wiki/Texture_mapping#Affine_texture_mapping