Reflection Personal Portfolio 2.4

Noah Brinkman – 515844

Contents

Introduction

I want to become a better gameplay programmer and game designer. Because of this I have chosen 3 skills I wish to exercise: Gameplay programming, Level design and Documentation. I believe these skills are important as they directly influence where I wish to end up as a technical designer.
To do this I will recreate a feature from popular game: Superliminal which uses optical illusions to solve puzzles, I will then create a level using this recreated feature and during the process of this I will document my progress as well as the code base I create.
Within this document you will find the research I did per leaning outcome, an explanation of my process as well as a reflection on each of the learning outcomes.
Then I will also reflect on the process in its entirety, log my hours and give a brief summary on the masterclasses I followed.

Learning Outcome 1

As an engineer I wish to increase my skill in being able to develop features and mechanics based on visual references and description as this might come in useful when working with clients that have abstract features in mind. To achieve this I will recreate a core feature from the game Super Liminal as shown in the following video. I will complete this feature before the end of the semester (week 4.8), the completion of which will be determined by my peer Kamilla Matuszak who will be given a small feature demo to determine whether the mechanic works on an acceptable standard.
Being able to recreate a feature based off of visual references is a useful skill as it could help me quickly breakdown gameplay (or a drawn storyboard) into features in a game to create. This allows me to efficiently work with other people to help them create features. I chose Superliminal's perspective feature as it is something I have never done before meaning I would go in with little to no prior knowledge.

Research

Forced Perspective

Upon observing the way, the game Superliminal works I can tell that it uses an optical illusion called forced perspective where we use the perspective of the camera to make objects appear a different size than they actually are. Take the following image for example:

Figure 1 https://sites.google.com/a/pgcps.org/erhs-photography/forced-perspective

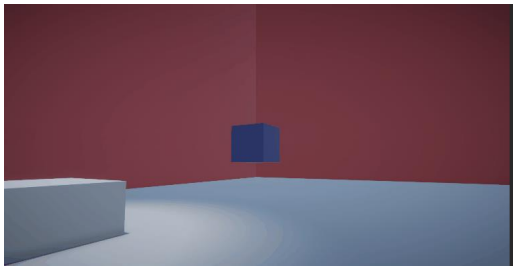
The man isn't a giant and the woman isn't miniscule. However, due to the cameras perspective it does seem like it. By abusing the way humans perceive perspective you can make very interesting optical illusions. But how does Superliminal use this?

How Superliminal uses forced perspective

Superliminal uses the forced perspective illusion by pushing objects as far back as they go and then scaling the objects to a size that makes them *appear* to be the same size as when you first picked them up. This is done by taking the distance between the camera and the object at the point of pickup (let's refer to this as SD) and the scale of the object at that point (referred to as SC). Then we push the object back as far as it will go and we then need to multiply SC by how much distance the object has travelled (if I pick up an object 2 meters away from me and it gets pushed 2 meters further away the object should double in size). So the formula would look something like this: New scale = SC * new distance travelled / SD. This relatively simple formula should do the trick.

Process

During my process I started with some minor research based on what the trick Superliminal does is called. I quickly came to realize that it is simply referred to as forced perspective. I then started working on how this would work best, looking for a formula that would easily and quickly make an object scale in the way I wanted it to. After some brainstorming and some more research I found a formula that seemed to do what I wanted it to do. After setting up a small unity prototype the result looked roughly like this:



Apart from the clipping issues there is also an issue that the object would be pushed into the walls. This issue took more effort than I would be willing to admit.

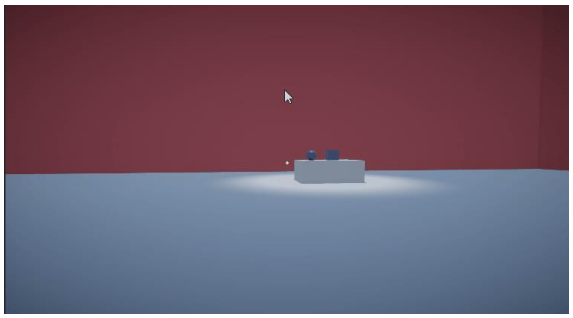
First I attempted to simply move the object back half its new scale in the direction of the camera, this seemed to work nicely *most* of the time. Until you would leave little to no space between the object and the camera, causing the object to launch the character into space again. I also did something wrong within the formula as the object would increase in size instantly upon the player picking the objects up.

These would break the game so it needed to be better. After a lot of tweaking within the formula I noticed that I used the current scale of the object (instead of using the distance it was when I picked up the object, I used the current distance) causing the object to increase in size the first time it is picked up.

For the clipping issues I looked at a method called “Physics.OverlapBox” which allows me to create (within data) a box that counts all the overlapping objects. This at first didn’t seem to do anything until I realized I already moved the object prior to setting the box, which was not smart of me.

Upon fixing this issue the objects seemed to behave rather nicely to this:

This is pretty much exactly the feature that I had in mind. After tweaking how much an object gets offset to get as close as possible without clipping through it. After this I added a quick outline shader to enhance visual clarity and my first learning outcome was done.



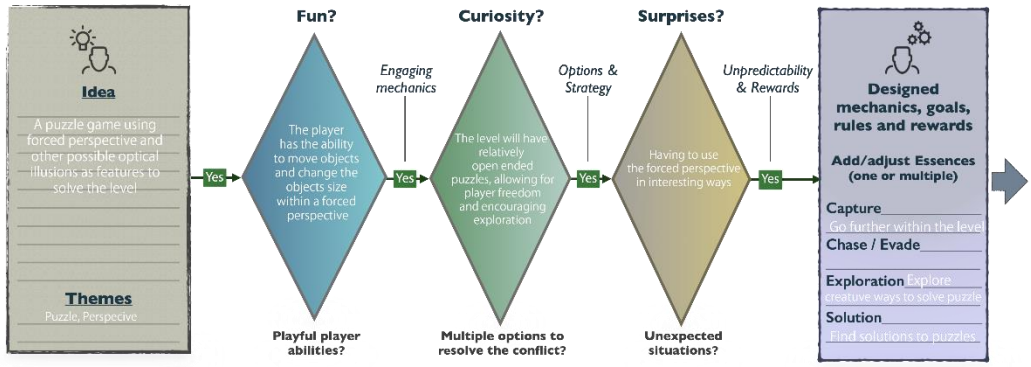
Reflection

Reflecting on the process, I am pretty happy how the learning outcome turned out. However, I do see some room for improvement. There is a decent amount of small mistakes I made that cost me some time In hindsight. If I researched possible pitfalls and resources, I could use within unity to help this effect I could have been don’t with this outcome a lot quicker allowing me more time for my other two learning outcomes.

Learning Outcome 2

As someone studying to become a technical designer, creating interesting and captivating gameplay is an important skill to improve. To hone this skill I will create one level of a puzzle game primarily using the feature from Learning outcome 1 before the end of this semester (Week 4.8). To make sure this level Is created properly I will consult the design principles and game design construct from Game design (Term 1.2). My peer Kamilla Matuszak will determine whether or not this designed level will be sufficient through means of a peer review. For this peer review I will extend the already existing peer review document as she will also need to be able to see my design decision In my process document to properly assess my designing methods and skills.
As a technical designer level design is something I would be directly involved in so practicing my craft on the design front is incredibly important as currently my particular branch of study has not had a lot of level design worked in, so I am lacking in this part of my craft.

Research



I started my research with a Game design Construct that allows me to have some insight about what the level should contain. I determined that apart from my currently existing feature of the forced perspective I should design interesting, unique, and open-ended ways of solving the puzzle. The primary way of advancing through the level would simply be going from point A to point B.

Process

When I looked at the empty unity scene that would become a level, I quickly realized I am out of my depth. I had a vague idea of “some” of the features I wanted, a few trippy effects that could ‘wow’ the player but nothing concrete. This is why I first started looking at researching and retouching up on game design Game Design Construct.
After the construct was created and I realized what it was the game required I got to work.
I started with a quick little tutorial, a room with a cube and a sphere the player can use to mess around with and the first obstacle, a door blocking the player from progressing. I then created a second puzzle, seemingly it’s a hallway with a ledge upward, you cannot normally climb onto it and the hallway is empty. The player should use one of the blocks they had in their possession to create a ramp upward. During the creation of this puzzle I quickly realized that my code from learning outcome one didn’t account for non uniform scale.
In layman’s terms, the scale (size) of an object is defined by 3 values X, Y and Z if all 3 of these values are the same the code would have no issue. Unfortunately rarely anything is perfectly sized on all axis, an easy fix I made for this was to create a parent object that had a scale of one on each side.

This would pretty much be used as a sort of multiplier for the object I *actually* wanted to scale. The issue with this however, is that I lost the properties of the original object. Which caused the object to phase into walls. After a lot of toying around with this I managed to get it working.
The puzzle was relatively straightforward after that, I made sure the player can push objects which allows for the player to resize the door push it over and now they have a ramp to walk on.
I then started theory crafting on the final puzzle, I wanted to make something a little visually interesting so in blender I made a hollow cube that I could rescale. The exit for the player would be in here, I am going to manipulate the players perception by using a plane on the mouth of the hallway. A plane within unity can only be seen from one side. Normally this is used to save triangles the player was never going to see, but it can also be used to create on way entries (if you know how to play with colliders). After making a start on that I decided to not jump ahead too far, I needed a second challenge. Something small and entertaining that allows the player to explore the concept a little more. I decided to make sure that objects would have correct mass in comparison to their size, meaning that we can also use a little bit of weight manipulation just like the size. For our second puzzle I created a pressure plate, this pressure plate will be linked to a weight value the player needs to hit if the player hits this the final puzzle piece will drop. The hallway (but too small to notice the hole at the end), I have decided to also link a little visual to the pressure plate so that the player knows if they're getting close. I decided to not make the game too difficult and make the 'sweet spot' pretty lenient. Otherwise the players might get frustrated before the final puzzle (that admittedly is a bit easier). For the final puzzle I at first decided that I wanted to just have the effect of walking into a seemingly hollow cube only to be transported to a long hallway that brings you to a congratulations screen. Instead I made it transport the player to a bit more fun of a location, an infinite corridor. How it works is there is a square hallway for the player. Only the player is stuck in one corner, turning the corner transports the player to the start of this corner. If the player moves backwards through the corner however, they will be allowed to pass.
This will lead the player to an actual fade to white followed by a congratulations screen.
The creation of the infinite corridor effect took a while to perfect, especially because I first attempted to brute force it. Partially out of laziness, partially out of arrogance and thinking it would be done easily.
After a lot of effort I decided to bite the bullet and start over (at least partially) by recreating the area to make it perfectly symmetrical, adding correctly placed lights to aid the optical illusion and adding some debug visuals to properly see where the player would end up after teleporting. After some more tweaking I got close to a perfect transition. It's not quite there yet but At this point of the process it has a lower priority than other tasks left on my to do list. After a quick testing round the game seemed finished and it was time to reflect.

Reflection

This learning outcome took a lot of effort especially as I quickly ran out of inspiration for interesting game ideas. It is unfortunate that due to the project I started as late as I did as I would've loved a little more time to work out the possibilities and put on inspiration for better gameplay.
I do think I ended up making an interesting enough level. Apart from that I thoroughly enjoyed the process. Thinking up interesting ways to trip up the players expectation of what happens next was a nice thought exercise for me and I had a lot of fun trying to think of ways, even if it came with a lot of struggles.
If I were to do this again I would probably try to start earlier and first come up with inspiration, then try to draw out or write down what I want to do before trying to make it within a game engine. I feel a continuous need for visible 'progress' which is a bad habit because it forces me to rush myself a lot. This is something I have been working on but still would need to find a way to resolve.

Learning Outcome 3

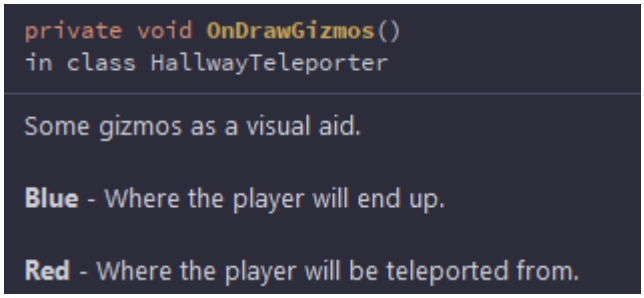
As an engineer, I wish to increase my ability to create qualitative, expandable, and readable code. An effective way to enhance this skill is to create concise and clear code comments summarizing my code. Before the end of the semester (Week 4.8) I will summarize and refactor the code I created within learning outcome 1 and 2 by researching correct and proper coding and commenting conventions. To assess whether this has actually Improved my codes quality, my peer Kamilla Matuszak will grant me a code review where we will determine whether the comments and naming conventions actually give the code more readability.
Documenting code in a proper and concise way makes code a lot more readable, expandable by other programmers and makes it a lot easier for yourself to remember what your old code does. This is a skill I have left behind in the past, despite it being a cornerstone of properly written code. This is why I added it as a (admittedly smaller) third learning outcome.

Research

Before I start commenting random thoughts, fun facts about the code, a properly written learning outcome for once or some rambles nobody understood I first have to find out how I can best document my code so that it will be up to professional standard. This means I first have to look at what *is* standard practice. I decided to look at what Microsoft themselves had to say about this. Microsoft learn is a website written by Microsoft to learn more about programming and the languages, tools and frameworks they create. I will use Microsoft learns commonly used code conventions as a reference for my documentation from now on. I will also take a look at the commenting documentation to properly be able to explain how my code works through within the text itself.
Through reading these web pages I can summarize that:
Comments should be descriptive with proper punctuation,
Comments should not be written on the same line as code and shouldn't be part of a commenting block (capsulated between/* and */) and Any method or class that needs explanation but especially public methods and classes should be summarized by adding a comment above them with <summary> and ending with </summary> any parameters that need explaining should be put underneath this with <param="the name"> and </param> this way you can describe each individual parameter.
With these conventions I will take another look at the project I created in learning outcomes 1 and 2 and I will apply the conventions to the best of my ability.

Process

With the conventions in mind I took another look at the project I created during learning outcomes 1 and 2 and realized I was lucky that the code base was relatively small meaning there is not a lot to document. This means that I have time to get more in depth and really document the proper way.
I first decided to look at naming conventions, usually this is based on preference for programmers, but it does need to be consistent. After a quick touch up on that I began the documentation process.



Mostly this was adding summaries to methods that could use it, explaining the math in layman's terms and then adding some nice formatting which was meant I had

to go back and forth between the commenting documentation a lot to make sure everything looked nice and professional for example: The following method was relatively small and served as a visual aid, this allows an engineer touching the code for the first time to quickly realize what each colour represents as well as what the method actually does. After the commenting I realized I had more than enough time to look at my naming conventions, which were all over the place. This meant that I had to rename all variables that didn't fit the norm.
This took a little bit of effort especially as it removed some of the values I set within Unity's inspector. After finding the "FormerlySerializedAs" attribute which allows unity to remember the values.
Upon making sure the game still works this process was finally done.

Reflection

This learning outcome took significantly less time, most time was done re-reading my code trying to make sense of it as it was a bit ambiguous at times meaning I had to carefully read it. I do think I should've done this learning outcome at the same time as my other two to really make sure I know how the workflow goes. But I do think I learned a bit from it.

Overall reflection

Overall I believe I finished all my learning outcomes to some extent at least. I think I did overestimate how much time my third learning outcome would take especially as I should've woven it between my first and second learning outcome. This is something I will have to take into account in the future.
I am very happy with the results, unfortunately there are still some bugs but after having a friend test the game a little bit I did have get some (verbal no official testing) positive feedback which makes me very excited to maybe continue the process and maybe create another level.
Looking back I do think I should've either started earlier, taking more breaks from the project to do work on the learning outcomes as I currently had to spend a lot of time back to back on this to try and finish the outcomes in time (as seen in the hour log a few pages down) this was very mentally draining and I do believe that is shown in some of the originality of the puzzles.
All in all I am very happy with my results and think it was well worth the effort but there are some pit falls with timing and scheduling I fall in a lot which is definitely something I will have to keep taking into account moving forward.

Masterclasses

Quick prototyping

During the quick prototyping masterclass, I attended a lecture by Yvens talking about how to quickly make prototyping tools within unity. Unfortunately, I knew a decent amount of the things Yvens talked about. He did however, make me interested in a quick prototyping tool/framework I could make that would allow for small games to be made with little to no code, primarily by using unity events to allow for a lot of things to happen without having to alter any code.

Matrix CMGT special

For the Matrix special we watched the Matrix movie in the cinema, there was also a special video created by our teachers replicating and parodying some of the most famous scenes from the trilogy. The experience was a lot of fun.

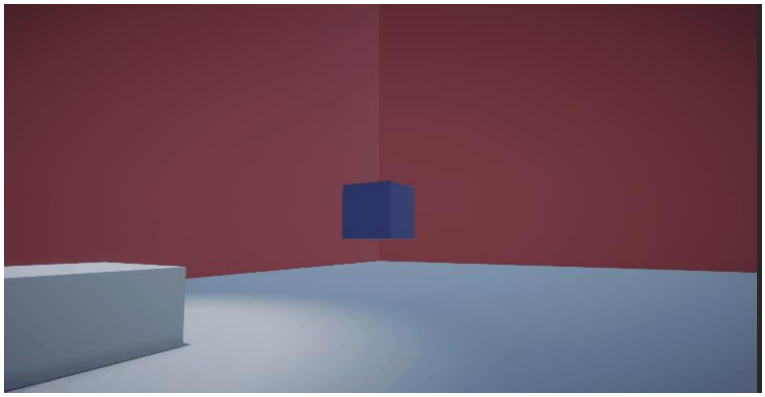
Hour Log

26-5: Setting up learning outcomes: 3 hour

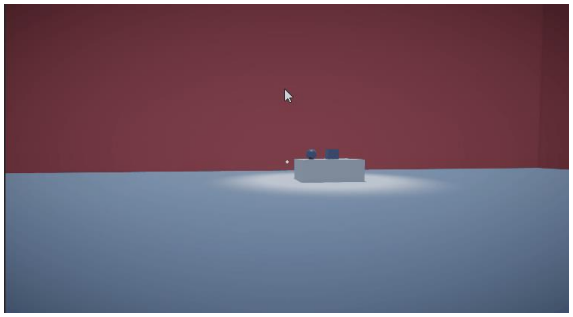
30-5: gained feedback, compiling and implementing: 1 hour

08-6: Researched forced perspective and started a small scene with a pickup mechanic: 6 hours

14-6: Tweaked the pickup mechanic and updated it so the size changes (incorrectly): 2 hours
15-6: Worked on the sizing, it's still incorrect but a noticeable improvement: 3 hours



20-6: Sizing still broken, works better though.: 2 hours
21-6: Sizing works a bit better, still would need some fixing: 1 hour
01-7: Sizing works near perfect, it flickers and sometimes crashes but progress: 3 hours
03-7: Sizing flickers stopped it works relatively well sizing up, sizing down seems to be a bit messy: 5 hours



04-7: Polished, fixed sizing down issue and added an outline shader.
Worked on the process and reflection: 6 hours

05-7: Finished process and reflection: 2 hours <-Outcome 1 done: total 30 hours

06-7: Research on game design and game design construct, gave me vision on what I needed to do, small tutorial: 5 hours.

07-7: Start on first puzzle, encountered a lot of bugs in original code. Did proper testing and bug fixing 4 hours.

08-7: Start on second puzzle, hyper focused on shader fix. Did some theory crafting for the final puzzle 6 hours

09-7: Second puzzle progress, finished effect (placeholders) for final puzzle all that's left is logic: 1 hour

10-7: Second puzzle finished, final puzzle almost finished. A lot of tweaking and debugging some writing: 10 hours.
11-7: Finishing up Learning outcome 3 and writing my process, outcome 3 reflection, masterclasses and overall reflection: 6 hours

11-7: Documenting Learning outcomes again, sending test out to Kamilla Matuszak.
Total: 65 hours