**Developer Diary**

***9th February 2018***

*17:00*

Started development with having a look through scripts that are inside ARKit unity plugin. No testing or changing code was done as of yet, just looking to through the scripts to get familiar with them and how they operate.

The goal for today was to start research and comparing more games which were implemented in AR, that use environment around player to work – how is UX designed in those games. VR games are to be considered too.

20:55

AR game titles have been looked at, where features and gameplay can be relevant to my project idea. Found out that at the moment there are not many titles that have relevant game mechanic, design that my project is focused on (world spawning around the player). Games looked at have minimalistic UI, have to figure out a clever way to minimalize UI in the game.

Would be helpful to look at some VR games, to find out how they indicate information around the player.

***10th February 2018***

*15:15*

Had some issues uploading files to GitHub, where progress from today was deleted and had to start from beginning. I have done what was recommended to me, creating a prototype which tests tracking of world even when the spawning location is not in camera view.

This was implemented using detecting horizontal plane, as it will work with any user input, however I thought of implementing some kind of board game into this game and having horizontal plane to have as a starting point sounds like a good idea.

Game involves having a cube in the spawn location which is initiated by the player, and then based on that location spawning two spheres, which have different locations. Player can easily view them and look at them without looking at the spawning location and it tracks very well. No issues.

There was a bit of learning required to get used to the code and how it would operate. After I got familiar with the code and how it operates, I didn’t run into any other issues.

*19:30*

There was an error with Developer Diary file, it was corrupted, had to re-download file that was backed up on GitHub.

Islands, which look like planets with very distinguishable colours for the time being, is implemented. First thoughts are that the planets are too close together, are a bit too big (size of 0.5 on all axes on Unity). For the initial implementation it is fine, will have to test it out and get feedback on the situation. Need to gather also feedback on the spawn location, if the plane is required or not – will depend on feedback and how player wants to play the game.

Created player character – a cylinder for initial implementation – where the top positions of each sphere are the locations that the player will be able to travel between. Not able to travel yet, but does spawn at the correct position.

There was a small challenge of getting the top of each sphere to be recognised as a spawn location, but it was overcome. The locations are stored in an array which can be easily accessed. At the moment, everything is being implemented in one document, which will change soon. It is done this way at the moment for testing purposes.

***11th February 2018***

*17:15*

Today focus on development was to get movement (tap on island to appear on that planet). This was challenging. Initially I tried to implement it using HitResult that is in ARKit plugin, however it didn’t work as after searching on the documentations, it appears that it only detects plains, and real-world objects and interactions, not what I was looking for.

My second approach was to use raycasts, which worked at the end, but there were problems to overcome along the way. Initial problem was getting 2D input to be transformed into the ray – it was quickly overcome by just double-checking code. One problem was detecting which planet it hit. As dependant on the planet hit, character will appear on that planet. Once I detected which planet was hit, the player appeared inside the planets – which isn’t visible – instead of the spawning locations.

It did work when I hard-coded the location as it had the same offset throughout every island, but I wanted it to work with the location points that I initialised every planet with. After further testing and changing things, I realised that it was the code that I saved the location points at that was a problem and the way that I accessed them. Now they do get written and read properly. There is an issue that the very planet (last in array) that is in the origin doesn’t get read properly, so have to investigate it.

*20:50*

The tap movement system works, for as long as the island/planet is not at the origin, it has offset of at least 2-axis from the spawn location. One solution that may overcome this issue is adding 0.1f to one axis (as it doesn’t work if position has the same 2-axis as the spawn location). Minor issue that can be worked around, and one case where it won’t work.

As the implementation is concerned at the moment, it is slow, as whenever I make any changes, I have to wait first for build to finish in unity, before building it on iPhone using Xcode which is a slow process, so testing is not as quick as I would have liked which affects the speed at which it is implemented.

The goal for today was achieved, where tapping does work to move the character. It has a bug that can be worked around as it is very minor condition which it works under.

***17th February 2018***

*14:15*

Started implementing Sprint Two today. Started with implementing testing buttons to have a prototype position where they might be. Full UI implementation will be done at later date. Today I started implementing Virus and first thing which I wanted to implement is virus showing up in game. This was done without many problems using Coroutines**.** When it was implemented there were some issues, but they were issues setting up (assigning materials again since I renamed the variable), which were easily fixed. Once they were fixed, the game plays like expected and I set it up to initially wait 90 seconds before virus shows up but it can be easily changed.

At the moment, the whole project is being implemented on one single script. When everything is working and being implemented – thinking at the end of Sprint Three – then I will split it up into separate scripts to be neat and easier to manage the code.

***18th February 2018***

*13:55*

Today I am focusing on visualisation of virus and the spread of the virus. I increased the spawning planet size to 10 – for better testing – and made the planets smaller (to 0.2f), will get feedback in testing to see if it is better to have the planets bigger or smaller.

I managed to implement spreading to the nearby planets without any troubles. I used mathematical Pythagoras equation to find the closest planet for which it will be infected. Once infected, at the moment it just changes colours and the planet tag changes from “*Planet*” to “*Infected*”, this information is used to find out if the planet is to be infected or if it’s already infected.

By testing it out, the spread of planets at the moment doesn’t look great and some planets are too close to each other, so once everything is implemented together, the plan of where each planet will be located – with the agreed size – will be implemented.

When was looking through the code, I found little bug that I fixed, where initially when I was calculating distance, I was only taking account x, y axis and y axis was taken into the equation twice, fixed it by changing one y axis to z axis.

*15:40*

For testing purposes, I reduced the time which takes for infection to start from 90.0f seconds to 20.0f, and also the time which it takes to spread is quicker than it will be implemented in the final product – 5.0f is set to at the moment.

The alert system is implemented – in portrait mode, have to get feedback which orientation the game should be implemented – when the infection begins. It starts 5.0f seconds before the actual infection appears on the planets with outline and text appearing on the screen. It looks ugly. But for testing purposes and gaining feedback, I think it will work. It will be changed later to look more pleasing.