**Game Specialism 1 – Programming (Inayah Iqbal)**

**Portfolio**

[programming brief projects - inayah](https://stulsbuac-my.sharepoint.com/:f:/g/personal/iqbali6_lsbu_ac_uk/EiCMQV_D-ZZFjhu4vsJpTwgBuoDJF8aQMAwZHStH8qNPtg?e=Z5nuWY)

(this OneDrive link can only be accessed by people inside LSBU. In the folder you will find **Unity packages** for each brief as well as an **assets file** in case the package isn’t working, which you can drag into unity)

**GitHub**

<https://github.com/aienae/rolling-road-final>

<https://github.com/aienae/waterfall>

<https://github.com/aienae/Speed-Dial-full-program->

(In these GitHub links you will also find each brief along with the assets folder and documentation)

**Reflective Statement**

The first brief I chose was ‘rolling road’, before I started I planned different ways I could approach this brief so it only focused on the main functionality. The aim was to make an infinite generating road so I researched some examples of this in use in games such as ‘Temple Run’ or ‘Subway Surfers’ that are well-known. I also looked at ways different people have re-created this online on sites like reddit or YouTube. After I had got down some main functionality, I ran into a few problems like when I wrote the spawning code, there was an error that created gaps in-between each plane of the road, so it didn’t look like a road, more like a broken road for parkour. As the game played for longer, the gaps started increasing in size. While I did find a few different tutorials tackling this problems in many different ways, I still couldn’t figure out why my solution wasn’t working. So I decided to discuss the problem with one of the TA’s at LSBU. We found out was the issue was the ‘origin’ of each plane, I had placed the centre of rotation at the start of the plane when it made more sense to place it at the end where the next plane should be spawning. While this fixed the issue of spawning the next plane, the code to delete the past 2 planes after the player passes them was not working. I used a ‘hitbox’ or collider which detected if the player had passed a certain section of the road, which would then delete the last 2 parts of the road behind the player. The error I made here was I didn’t declare my variables correctly and I didn’t link my code to use the hitbox to delete the road, I just wrote the code to delete the road with no conditions on when to do it. I only noticed this when I got someone to peer review my code. Thanks to this I realised I should peer review my code more often because it helped me understand how other people interpret my code which helped me notice my own mistakes. In terms of what I think I did well, I think my research skills really helped me complete the brief, I had to look through many forums on the unity website as well as on other sites like Stack Overflow. I also think I improved on thinking more logically and getting used to the C# language as it was my first time properly building something like this and using this language.

My second brief was the indexed scrolling texture. I was excited to do this brief because I got to think about how I could implement the texture in a real life ‘game’ situation, so I decided to use a waterfall as an example. Again before I started I did my research on how this could be done in unity and watched tutorials to help guide me. I didn’t run in many problems, the only thing I needed to consider was finding a texture that would look good when being tiled so I tested many different ones until I found one that I liked. Something I could have improved was the layout of the project felt a bit plain so I could have spent a little more time applying the texture to an actual waterfall object to make it feel more alive. As for what I did well I was able to solve all the problems I had independently so I was definitely improving a lot more from my previous project.

My third and final brief was the speed-dial. This brief was probably the most challenging for me as I had to use the UI features of unity which took some time to get used to because I didn’t fully understand them before I started. I struggled a bit thinking about how I could demonstrate the use of the speed-dial without making the project too big or too optimistic. So I had the idea to use a bouncing ball for speed dial to measure its speed. This turned out to be so much harder than I had anticipated. I made loads of small mistakes which confused me because everything seemed like it should be working. One of them being the ball would bounce extremely slowly, which isn’t ideal for the purpose of the program. I tried changing the weight of the ball, making it lighter or heavier, even doing the same to the platform it was bouncing off of, but it still made no difference. I was using the unity ‘bounciness’ physics so there isn’t much that could go wrong. After maybe an hour or so of research, I soon found out the reason was because my ball was massive. The scale was completely wrong. I didn’t know that this could affect the balls bounciness, so when I scaled everything down it was bouncing like normal. This was very frustrating to me because the solution was actually very simple and I had just missed it. But that was just one of many issues I encountered. Another issue was creating the arrow for the speed dial, the centre of rotation was not working in my favour and kept getting skewed even when I changed the origin’s position. A solution I found was to create an empty and use that as an anchor to rotate the arrow the way I wanted. It sorted out the issue instantly and I wish I had known it earlier. Overall I learnt a lot about attention to detail on little things and principles of unity I didn’t know before. As for what I did well, I managed to come up with the easiest solution to solve this problem, it was effective and demonstrates my speed-dials functionality well in a simple way.