**Programming Specialism**  
**FPS Counter Brief Documentation**<https://github.com/JPLDN/FPS-Counter-Brief-1.1> - Brief Link 1 **A screenshot of a computer

Description automatically generated**- A scene I made and adding in particle systems, by going onto the hierarchy, right clicking, and selecting ‘Effects’ and selected ‘Particle System’ to test out how the FPS is measured and how it fluctuates and changes.  
  
A screen shot of a computer program

Description automatically generated  
  
- A script I used for the FPS counter to measure and calculate the FPS when the particles are being spewed in the Scene.   
  
A screenshot of a computer

Description automatically generated  
  
- I made a UI Canvas and added in Text, to which I then added my script into the Text so the FPS can be read and show the changes and measures of FPS.   
- In the script I posted earlier, I also added and extra element where I must “show the frame rate in milliseconds per frame as well as frames per second.”   
- That is why you see “0m/s” underneath “0 FPS”.  
  
A screenshot of a computer

Description automatically generated  
  
- Updated the particle systems, adding in explosions of particles, with this, whenever the particles explode, the FPS drops then rises back up when the explosion is no longer occurring, but when it does occur again, FPS drops.

**Shuffle Brief 2 Programming Specialism Documentation**<https://github.com/JPLDN/Shuffle-Brief>- Brief Link 2 **A computer screen shot of a blue screen

Description automatically generated**  
Firstly, I made a scene in Unity to use it as a test scene to see if the Shuffle script works.   
  
I firstly created an empty object and named it ShuffleTest, this is so I can attach my script onto something to have the Shuffle function work.   
  
I open the scripts folder and create a C# script, I then began to experiment and test out different scripts as well as using tutorials to help me get to what I was looking for.   
  
**A screenshot of a computer

Description automatically generated**  
  
Here's what the finalised script looks like after experimenting and trying out different ways of getting the shuffle function to work. This was done with the C# script that was made at the start when I initially created the Unity test scene.  
  
**A close-up of a black background

Description automatically generated  
A black and white background

Description automatically generated with medium confidence**  
  
Visual example of the shuffle function, without the numbers repeating twice in row.   
  
**A screenshot of a computer

Description automatically generated**  
  
The script was also adjusted to be able to be edited in the inspector, I then named 10 tracks since I found it to be a good amount to test the shuffle function.

**Name Generator Brief 3 – Documentation**

<https://github.com/JPLDN/NameGenerator-Brief> - Brief Link 3 **A screenshot of a computer

Description automatically generated**To start, I created a new Unity scene to use for the Name Generator. The first thing I did was create empty objects and renamed them. Initially, I had 2 GameObjects, one for the actual name generator and one so it appears on the console, but instead, I changed that and decided to put the debug log onto the name generator script as well.   
  
A screenshot of a computer program

Description automatically generated  
  
This is my first initial attempt at writing the name generator script. It took a few attempts and trial & errors to get to this as well as help from others to get it to function. I then made changes and additions to it later such as adding numbers to the script as well as it being randomised for names to have numbers and to not have numbers, and if the names do have numbers, it’s randomised to whether they have 2 digits or 3 digits in the name.   
  
A computer screen shot of text

Description automatically generated  
  
here's what the main part of the script looks like that generates the names with the numbers being randomised as well as how many digits being randomised when being generated.   
  
A screenshot of a computer

Description automatically generated  
  
Here are some examples of the name generation, generating names both with and without numbers.   
  
A screenshot of a computer

Description automatically generated

And these are examples of the names that do generate numbers, either generate numbers with 2 digits or 3 digits.

**Reflection Document**

I think that the briefs went well, and I learnt a lot of overtime whilst working on and completing the briefs. I think the things that went well were that I managed to learn a lot about programming and the different types of code for programming. The first brief I chose was the FPS Counter, I chose this because, in all games, there’s an FPS counter setting to check the frames per second when turned on, so naturally I felt more interested to see how an FPS counter is coded and how it works when in testing. Firstly, I opened a scene in Unity to begin, creating an Empty new Object as well as creating a C# script, of course creating different files i.e. scenes and scripts to allocate everything to its correct file. Once that was done, I then began coding for the FPS counter, it took a few attempts with online research and tutorials to then get it to where it needs to be and function correctly, I then decided to add particle bombs into the Unity scenes to test out the FPS counter. I tested it out and it turned out to be working fine, so to further test the FPS, I added even more particle explosions, of different sources and settings, some being a particle explosion, some simply being a particle generator. Once it turned out how I wanted, I then looked at the little bonus credits for the FPS Counter. The extra credit I ended up doing for the FPS Counter was “showing the frame rates in milliseconds per frame and frames per second”, it went well considering that the code for frames per second is pretty much the same or similar except its recording milliseconds not seconds. For the second brief, I decided to choose the Shuffle brief. The reason why I chose the shuffle brief was because I think it’s something that I don’t think many people who be too interested in so learning something that perhaps not many people know/interested in I think would help me a lot, especially with how broad coding, C#, can be. Of course, I first opened a Unity Test scene to get started with my Shuffle Brief. The first thing I did was created those dedicated files (scripts and scenes) to allocate the things I need to the correct place without it being so clustered. I then began to work on the Shuffle by creating an Empty Object and created a script to begin coding the shuffle mechanic. I started off with looking at how a shuffle mechanic works via the explanation on the brief website where you choose your briefs. This gave me a better understanding on how the script for the Shuffle mechanic would work, I did this because I was kind of stuck with how I should do this, so I resorted to looking at tutorials and doing research on how the shuffle mechanic functions. Once I had a good understanding of it, I then began coding the script for the shuffle mechanic. This did take a quite a few trials and errors, mainly because the code had errors or just didn’t function how I wanted it to. Once, I had finally gotten the shuffle mechanic to work, I needed to make sure the numbers (examples) weren’t repeating and that it would shuffle randomly. This also took a bit to figure out, so I asked someone to help me with how to prevent any numbers from repeating whilst also randomising the shuffle every attempt. I also needed to make sure that the numbers were also accessible and can be edited in the inspector tab in Unity which was simply done via the use of a public variable. For the 3rd and final brief, I decided to choose the Name Generator, one of the reasons to doing this is that I wanted a pattern of choosing Beginner, Intermediate then Advanced, initially I was struggling to choose what brief to do for my 3rd brief, so I reached out to a friend to help me decide. In the end, I chose Name Generator since the coding I was familiar with, was more so towards game programming rather than coding as a whole, I learnt that for name generation it involves algorithm work, to which I didn’t know much about. I did a lot of research and looked at tutorials to help me learn about algorithm works, and once I got to a good understanding of how it works, I attempted to code the script for a name generator. Of course, starting off with a new scene in Unity and then created 2 Empty Objects, one for the actual script itself, the other simply for debugging, which later was removed and put into the main script. This was the brief I struggled most on since it required me to do multiple trial and errors, as well as using tutorials and asking for help to understand how name generating works. Not only did I have to create a name generation, I had to make sure it wasn’t duplicating names as well as avoiding obscenities, which also in turn took a bit of time, but not as long as the actual name generation, since I had to provide a few examples of obscenities being blocked off and showing it works and avoids those name generations, which can also be edited in the inspector. Overall, I think the briefs went well and I learnt a lot with the completion of these briefs, I liked how I was able to do a lot of research and learn about new things about C# since I was aware with how broad coding is but didn’t know fully how broad and versatile it really is. I think I could’ve worked on my time management better for these briefs as well as asking for help a bit more since me keeping my struggles to myself won’t get me anywhere.