Names: Scott Cooper, Lucas Harvey, Kenneth Ho

100580683 100585553 100583602

**Q1**

For assignment 2, we decided to go with the particles assignment over the squash & stretch animation assignment. The assignment was built off of the tutorial code from lab 5. The completed project demonstrates 3 different uses for particles. The first emitter creates a trail of particles accelerating downwards. The second emitter shoots particles straight upwards, which promptly explode similarly to fireworks. These fireworks have random nodes that make them spawn in random locations and, once they explode, the colour of the fireworks are randomly generated as well. The last type of particle generated is a confetti explosion which also has random colour and location generation.

There are many reasons for choosing these ways of emitting particles. First of all, the particles can be used to designate game functions visually to aid the player. For instance, the particle trail can be used to demonstrate functions such as increases in speed. In addition, the fireworks can be used at the end of a game to congratulate the winner and the confetti can be used to demonstrate a spark or another effect for celebration. The particle effects help convey visual information to the player and can be used to provide positive feedback so they know they completed objectives correctly. Furthermore, the random colour generation for the fireworks and confetti add variety and are aesthetically pleasing. Another reason for deciding to create these particles is that they can be used universally in many games. Trailing particles and fireworks/confetti can be applied to many games. These effects do not have a singular purpose and can be altered for different situations. Lastly, these particles are simple and allow easy modification in the code to make them more dynamic and customizable.

**Q2**

Create emitter (one of 3 classes: firework, confetti or path)

Initialize position

Initialize velocity

Set velocity limit

Initialize colour

Initialize Acceleration

Spawn emitter

Update attributes\*

Wait for input

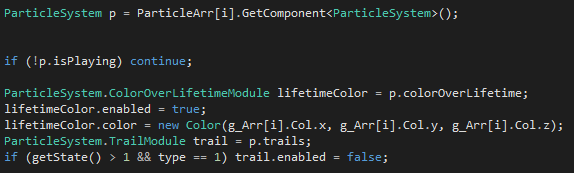
Toggle emitter types

\*Attributes = position, velocity, colour, acceleration, etc.

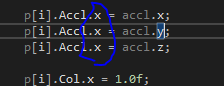
\*Depending on the emitter class, particle system will update attributes differently

**Q3**

To begin with, one of the biggest problems that we encountered was being able to overwrite the particle positions in Unity. The problem occurred within the SetPosition() massive for loop. Throughout the entire for loop, to be able to modify the particles in any way you need to get components from the particle system. This includes modifying: lifetime, colour, position, length, etc.

 Above is an example of the specific components needed to be gotten and modified, which took some time to figure out.

Another problem lies in our DLL code. Certain variables, such as acceleration, were not affecting anything within the code. The variables were not altering the behaviors of the particles correctly and would not function as intended. This issue happened because we accidentally had multiple x values instead of x, y and z values. Changing this allowed the variables to work properly.

 This was the issue.