# Complex Games Evaluation

# Particle System C++

# Issues

a major issue encountered was not being able to render both the fire and smoke particles which was caused by having only one initialise class to share the particles. Which resulted with the draw function rendering the second particles as the update would override the first update particles in the VBO. Which was fixed by making another initialise class, so the fire and smoke particles didn’t share the same VBO. Textures having black background with the sprite that particles that overlap other particles would see the background cutting off the particles behind and not blending. This led to coding in Alpha blending in to make the background transparent and the smoke particles get thicker in alpha when overlap and fire particles get brighter in colour when overlap. In the EntryPoint.cpp I wrote glBlendFunc with GL\_ONE GL\_ONE\_MINUS\_SRC\_ALPHA in the parameter to have a permultiplied alpha blending for all particles which got rid of the black background. However, the fire particles needed to be additive alpha blending. This would be done in the fragment shader with depending on the value of alpha (vec4 alphaAge.x) which was change in the fire compute shader setting it to 0.0f as the default was 0.1f. If greater than 0.5f run the statement for smoke permulitplied alpha blending, else run the statement for fire additive alpha blending. This solved the issue of the black background and sorted the alpha blending to the right particles. In the compute shader coding the velocity of the particle I found using gravity to be a big problem to work with as it would cause the particles x and z values of position to equal zero. This meant that the particles would start spread out but as the project kept running it would cause the particles to stop spreading out but only up on the y position till it was a line shooting up of particles. To stop this from happening I change slowed downed the velocity after it was applied to the position and updated with the new acceleration by multiplying by 0.999. I also add a new vec4 newVelocity into the particle struct that would grab a random number between -0.5 and 0.5 for the x and z. that would be called in the compute shader when the particles age hits zero to reset the current velocity values. I had to change glBindBuffer(GL\_ARRAY\_BUFFER, particleVAO) to glBindBufferBase(GL\_SHADER\_STORAGE\_BUFFER, 0, particleVAO). This was causing the update to not render the data stored in the ssbo because I had the wrong target in the parameter and was using the wrong function. The change fixed the problem and render both compute shaders with the right data selected.

# Performance

The size of the particles came an issue when trying to render 1,048,576 would be too much for the project to handle and would easy slow the rendering. The size I went with was 16,384 was I got from 128 the work group size the compute shader would take to the power of 2. This showed the project running smoothly with this number of particles. Another performance I try was initialise an array of 20 classes of the particles and had it run through the update in a for loop that called the update and draw function. This resulted in a white screen for a second than the particles rendering after. The number in the array that seem to not have a second to load with a white screen was 9 particles being initialise at the same time.

# Changes

A change I made to the project was to add billboarding to the particles which involved added another shader the geometry shader. This meant I had to link the shader to the vertex and fragment shader which is processed in between the two shaders. I had to rewrite the in and out of variables to go from vertex to geometry to fragment shader. Then fill out the geometry shader with the correct code so the particles will always be facing the camera no matter where the camera is place or looking. I Added alpha blending to the project which this would get rid of the unnecessary colours in the image of the particles and look more like fire and smoke when layered over another particle. I original going to do all the different types of particles in one compute shader but after working on it found that using to different compute shaders would be a lot easier to manage getting the right data to the right variables than of struggling with one compute shader I couldn’t work with properly.