Members:
Chen, Vivian Angel
Elomina, Zivv
Naranjo, Johann

Github Link:
https://github.com/ElominaZivv/GDPHYSX/tree/Phase1

Implementation:

Requirements	Implementation
Particles	
Random Colors	There is a randomizer function which handles the randomizing. Then in the Object class there is a function that calls the randomizer function with the parameters of inclusive min and max. This color randomizer function is then called in the constructor itself so that the spheres will have a randomized color the moment it i s created
Spawns user defined number of “sparks”	Outside the while loop, users are asked to input an int num for the amount of sparks that should appear on the window. An empty vector then was made to contain the sparks. A for loop was made so that when i is less than the user input it will make a new sphere and push it into the empty vector, which will be filled up to the user indicated amount.
Spawn at an elevated point in the bottom	We set the position at -700 for all sphere For it to spawn sequentially (or not all at once), we delay the pushing of the new sphere into the vector that was mentioned in the section right above this one (Spawns user defined number of “sparks”).
Radius should vary from 2 - 10m	Again like the random colors, there is a randomizer function which gets called in the randomize radius function.

	It also gets called in the constructor so that the size will be set the moment the spark is made
Damping	The damping we used was the one that was taught to us. It is a variable in the particle that gets update in the update
Gravity	The gravity we used is also the same one that has been taught. It is directly applied in the object world
Lifespan 1 - 10 sec	Like the random color and radius, this also utilizes the randomize function. This, again like the other, gets called in the constructor to ensure each spark has a randomized lifespan the moment it is created.
Play and Pause	A global bool was made so that when it is paused (bool == true), the world.update under fixed update will not be called.
Camera	
WASD	Inside the Camera class is float thetaX and float thetaY that store the value of the rotation of the camera around the fireworks. Every update, the camera waits for the user input. When the W key is pressed, the fireworks rotate away from the camera. When the S key is pressed, the fireworks rotate toward the camera. When the A key is pressed, the fireworks rotate to the left. When the D key is pressed the fireworks rotate to the right.
Movement	Given the distance of the camera from the firework, a triangle can be formed between the position of the camera, firework, and the ground. From that triangle, the movement of the camera can be computed to create the third person camera movement.

References:
Thin Matrix. (Nov 16, 2024). OpenGL 3D Game Tutorial 19: 3rd Person Camera. https://www.youtube.com/watch?v=PoxDDZmctnU