Nicholas Robinson

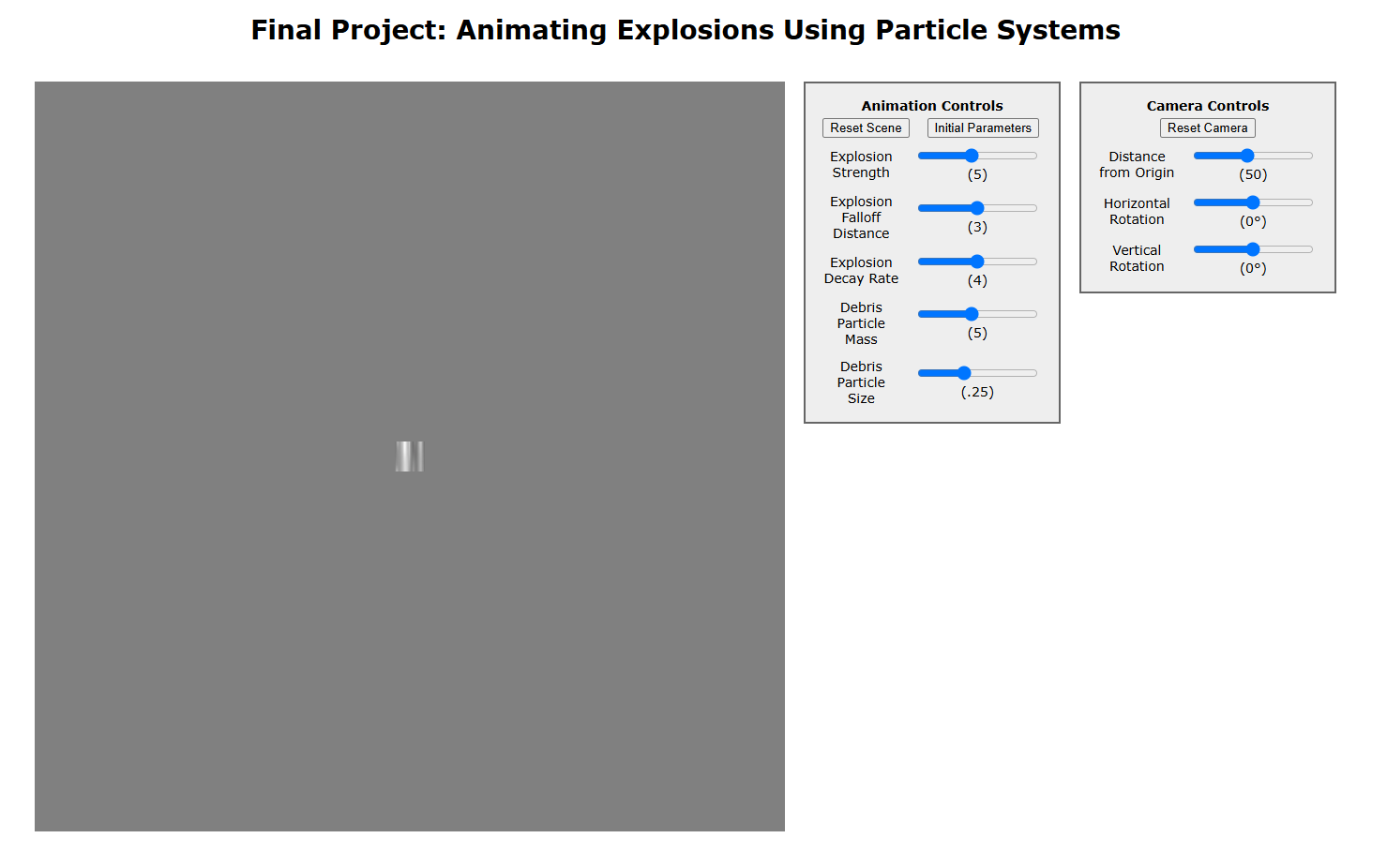
3/20/2025

**Final Report for Animating an Explosion using Particle Systems**

For my final project for this class, I created a HTML file called “Explosion\_FinalProject.html” that implements an explosion using particle systems. **Note that the textures for the particle systems will only load if the HTML file is opened in a local server, probably use the Live Server extension in Visual Code Studio and right click the HTML file to “Open with Live Server”**. This project was implemented using a combination of JavaScript, CSS, and HTML. The GUI design was partially inspired by the GUI from Project 1. Meanwhile, the 3D computer graphics was implemented using a powerful Javascript library called Three.js, which uses WebGL to create particle systems. Here are the links that I used to help learn how to use Three.js:

* <https://blog.pixelfreestudio.com/how-to-create-3d-particle-effects-with-webgl/>
* <https://blog.pixelfreestudio.com/how-to-create-3d-animations-with-three-js/>

In terms of the GUI, here is an image of the final project without interacting with the program:



For the formatting of the tables themselves, this is done in a file called “FinalProjectTable.css” in the “css” folder. In terms of the manipulation of the sliders within the tables and the corresponding variables in the code, this was done in the “GUI\_function.js” file. Note that for any of the sliders in the code, the new values will NOT change the behavior if in the middle of an explosion.

In terms of the Animation Controls table, here are the specifics:

* Reset Scene Button: Resets the scene only if the cube has exploded.
  + Deletes the debris, smoke, and fireball particle systems of the explosion.
  + Adds the unexploded textured cube back into the scene.
* Initial Parameters Button: Sets the animation parameters back to their initial values at the start of the code.
  + Will change the value of the slider, the sliders themselves, and the corresponding variables that apply to each of the sliders.
* Explosion Strength: Determines how big the explosion is
  + 1-10 scale with intervals of 1, initially set to 5.
  + Affects all three particle systems of the explosion.
  + Changes the “explosionStrength” variable in “particles.js”.
* Explosion Falloff Distance: The distance from the center of the explosion when the force of the explosion stops acting on the debris.
  + 2-4 scale with intervals of 0.25, initially set to 3
  + Only affects the debris particle system
  + Changes the “falloffDistance” variable in “particles.js”
* Explosion Decay Rate: The rate that the fireball particles will decelerate from their initial velocity.
  + 3-5 scale with intervals of 0.25, initially set to 4.
  + Only affects the fireball particle system.
  + Changes the “fireballDecayRate” variable in “particles.js”.
* Debris Particle Mass: Determines the mass of the debris particles.
  + 1-10 scale with intervals of 0.25, initially set to 5.
  + Only affects the force of gravity on the debris particles.
  + Changes the “particleMass” variable in “particles.js”.
* Debris Particle Size: Determines the size of the debris particles
  + 0.1-0.5 scale with intervals of 0.01, initially set to 0.25.
  + Only determines the size of the debris particles, with no change to forces applied to the debris particle system or its mass.
  + Changes the “particleSize” variable in “particles.js”.

In terms of the Camera Controls table, here are specifics:

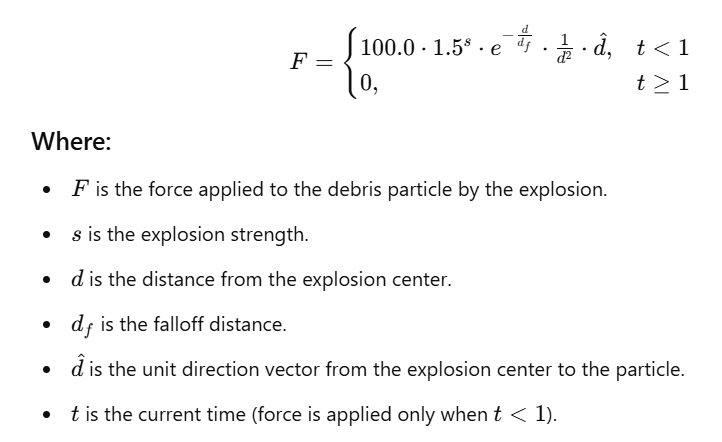
* Distance from Origin: Determines the distance toward/away from the origin in the z direction relative to the camera.
  + 10-100 scale with intervals of 5, initially set to 50.
  + Changes the “cameraDistance” variable in “particles.js”
  + Works even when the cube is currently/has exploded.
* Horizontal Rotation: The degrees the camera rotates in the x direction relative to the camera about the origin.
  + -45 to 45 degree scale with intervals of 5 degrees, initially 0 degrees.
  + The camera will still be the same distance from the origin after rotation.
  + Changes the “horizontalRotation” variable in “particles.js”.
  + Works even when the cube is currently/has exploded.
* Vertical Rotation: The degrees the camera rotates in the y direction relative to the camera about the origin.
  + -45 to 45 degree scale with intervals of 5 degrees, initially 0 degrees.
  + The camera will still be the same distance from the origin after rotation.
  + Changes the “verticalRotation” variable in “particles.js”.
  + Works even when the cube is currently/has exploded.

For the 3D graphics, the scene is initially a simple 3 by 3 by 3 cube mesh that is textured with a simple metal texture “MetalTexture.jpg”. **When the user clicks on the cube mesh,** the cube will start the explosion effect that is the focus of the project. On clicking the cube mesh, the following will happen:

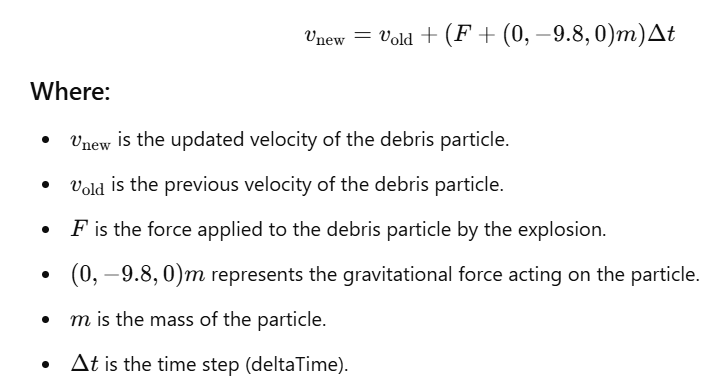
* Creates the particle systems for the debris, fireball, and smoke of the explosion effect and adds them to the scene.
* Removes the old cube mesh from the scene.
* Animates the particles in each particle system per animation step.
  + Done in real time, not with a constant time step.
* This will only happen if the cube mesh is in the scene, with the only way to do it again is to reset the scene.

In terms of the debris particle system:

* Creates miniature cubes that represent the debris particles in a cube formation to match the cube mesh, with the side length of said miniature cube particles being equal to the particleSize.
* The number of debris cube particles that are created is (⌈3.0 / particleSize​⌉)^3 (to the ceiling)
  + Since it is raised to the ceiling, if the particleSize does not evenly divide the cube side length of 3, then the debris cube will be slightly bigger than the original cube mesh.
* At the start of the explosion, the initial velocity of all the cube is set to (0, 0, 0).
* During each animation step, the debris particles will be accelerated at at a constant rate of -9.8 \* particleMass in the negative y direction relative to the scene’s coordinate system.
  + Debris particles are not affected by drag or air resistance
  + Represents the force of gravity acting on the debris particles.
* The force that the explosion applies to each debris particle per animation time step is relative to the position of the particle relative to the center of the explosion (the origin), the explosion strength, and the falloff distance.
  + The direction of the force is based on the position of the particle relative to the explosion.
  + Formula for the force that the explosion acts on the particle every animation step (note that the force from the explosion is only applied for the first second after the explosion):



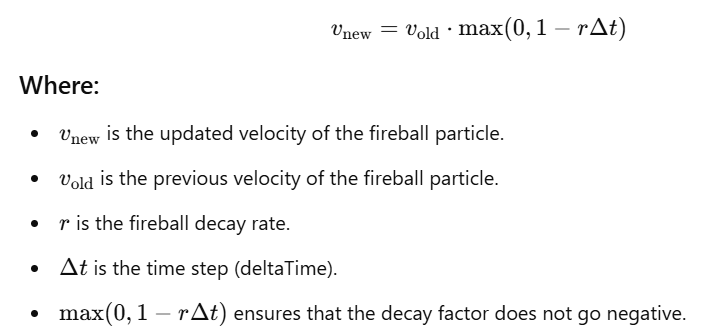
* Using this explosion force F, here is new velocity every time step deltaTime:



* + After 1 second, the particle is only affected by gravity and will have a constant velocity in the x and z direction.
* The new position of the debris cube particle at a given time step deltaTime is just the old position plus the product of the new velocity times deltaTime.

In terms of the fireball particle system:

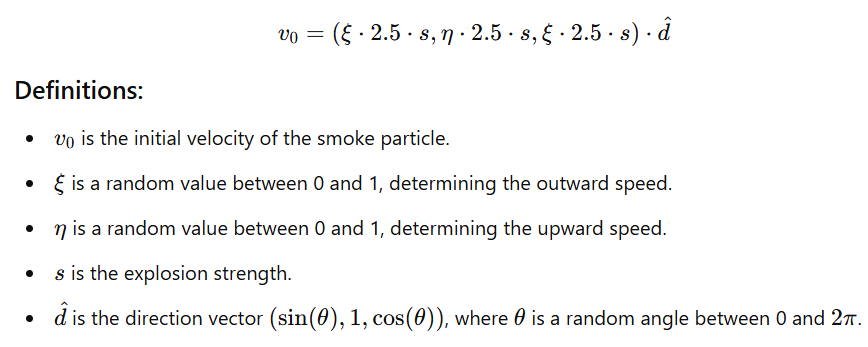
* Number of fireball particles is equal to 10000 \* explosionStrength.
* Each particle starts in the center of the explosion, then moves in a random direction from the center.
* All particles have a fireball texture “FireballTexture.jpg”.
* Each particle emits light and dims by this formula:
  + e^(-2t), where e is the natural exponential function and t is the current time since the start of the explosion.
* The particles are initially fully opaque and become less opaque over time through this formula until it is a low enough value that the particles disappear:
  + e^(-t\*4), where e is the natural exponential function and t is the current time since the start of the explosion.
* The particles start at a white color (1,1,1) and eventually transition to a red color (1,0.5,0) through this formula and is constant after 0.2 seconds:
  + currentColor = (1, (1-t) + (0.5 \* t), (1-t)), where t is the current time since the start of the explosion and t < 0.2 seconds.
* The initial velocity of the fireball is equal to 20 \* explosionStrength, with the fireball particles at the start being very fast, then quickly decelerating.
* The new velocity at each time step deltaTime is:



* The new position of the fireball particle at a given time step deltaTime is just the old position plus the product of the new velocity times deltaTime.

In terms of the smoke particle system:

* The smoke particle size is equal to 3.0 \* explosionStrength.
* The number of smoke particles is equal to explosionStrength \* 50.
* Each smoke particle has a smoke texture “SmokeTexture.png”.
* Each smoke particle initially starts in the center of the explosion.
* The smoke particles are transparent.
* Each smoke particle will always go in the positive y direction, but will go in a random x and z direction.
* Each smoke particle is given a random grey value between 0.4-0.5.
* The initial velocity of each smoke particle is equal to:



* The velocity is constant in the x and z directions, while the velocity in the y directions will change by 0.05 times deltaTime each time stamp of deltaTime.
* The opacity of each particle is initially 0.2, and slowly fades away at a rate of 0.02 times deltaTime per time stamp of deltaTime.

The hardest part of this project was simply getting started. Unlike my final project from the previous class, this one required me to start from scratch. While Project 1 provided some guidance with HTML formatting, I had to learn everything else on my own. I began by developing the program’s initial GUI, which took far longer than expected due to my lack of experience with HTML, CSS, and JavaScript. However, once I successfully added a single slider to the first table, the rest became relatively straightforward.

Another major challenge was implementing the particle systems of the explosion. I had never worked with particle systems before, so aside from the links I shared earlier, I had to figure it out on my own. Although I had a clear vision of the final result, achieving it required extensive trial and error. Each particle system needed precise equations to produce the desired effects, which took considerable time and adjustments. However, through persistence, I was able to refine them and achieve the outcome I envisioned.