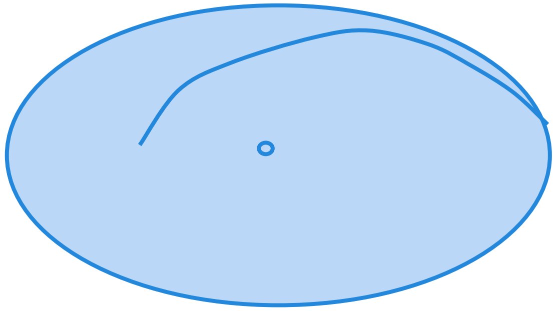
Team 6

Ajayveer Aujla
Ahmed Dorias
Ian Kelley
Simon Labute
Mark Massoud

Space Shooter 5000

Camera controls and alternate views

- Scenic camera
  - Third person view
  - Elliptical orbit



$$\vec{r}(t) = \langle a * \cos(t), 1.5 + \sin(t + \phi), b * \sin(t) \rangle$$

Left to Implement:

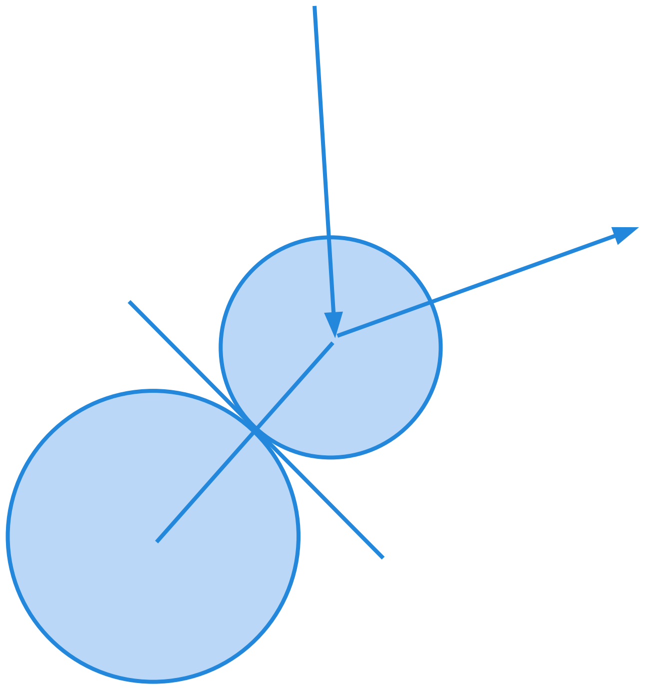
1. Randomize orbits
2. Curve into ellipse
3. Object Awareness

Collision Detection

O(n^2) pairings, so had to control number of models.

Collision occurs if:

$$||P_1 - P_2|| \leq r_1 + r_2$$



Physics Based Collisions

Equations to Satisfy:

$$M_1V_{1i} + M_2V_{2i} = M_1V_{1f} + M_2V_{2f}$$

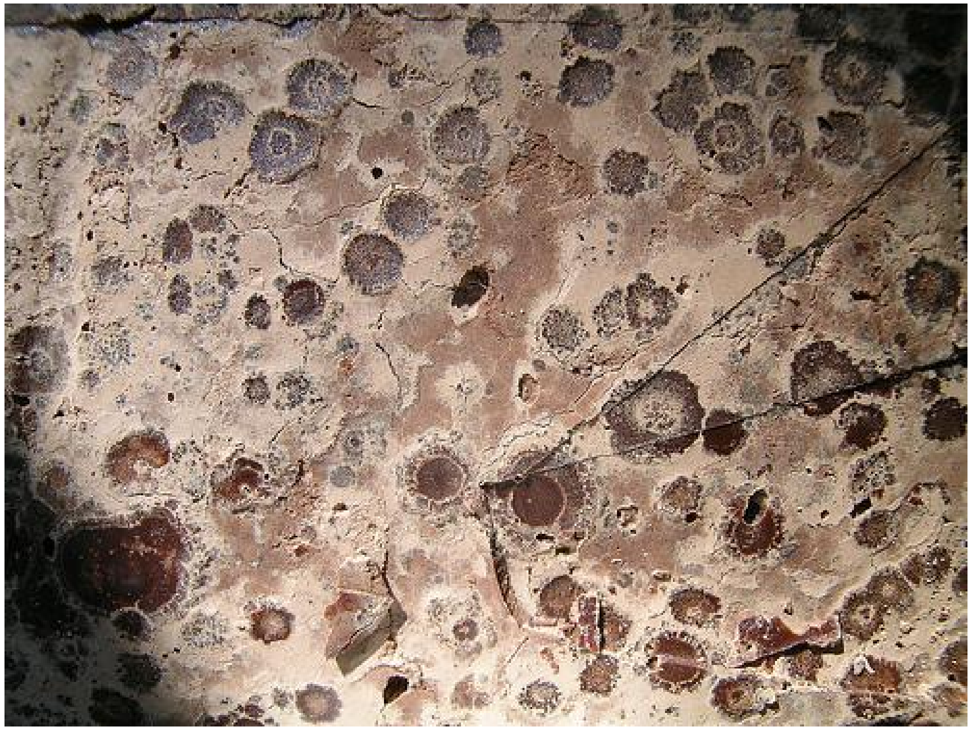
$$\frac{1}{2}M_1\|V_{1i}\|^2 + \frac{1}{2}M_2\|V_{2i}\|^2 = \frac{1}{2}M_1\left\|V_{1f}\right\|^2 + \frac{1}{2}M_2\left\|V_{2f}\right\|^2$$

Tasks in Progress/Wish List

- Inelastic Collisions
- Soft-Body/Deformable Objects (Not Started)
- Object Loading - Not integrated
- Gravity Processes

Asteroids...

Asteroids are inheriting from SphereModel, adding Texture



making:

B-Splines for trajectory

...and AsteroidSystem

- Logic similar to ParticleSystem
  - generated every X seconds
  - destroyed once they hit world origin
- World has an instance of AsteroidSystem
  - updates it

Particles

- Implemented smoke particles to follow the asteroids.
- Created particles to fall off the asteroids once hit
- Added a geometry shader on top of the other shaders.

Miscellaneous Upgrades

- Multiple Light Sources
  - Modified vertex and fragment shader to implement multiple lights in the scene.

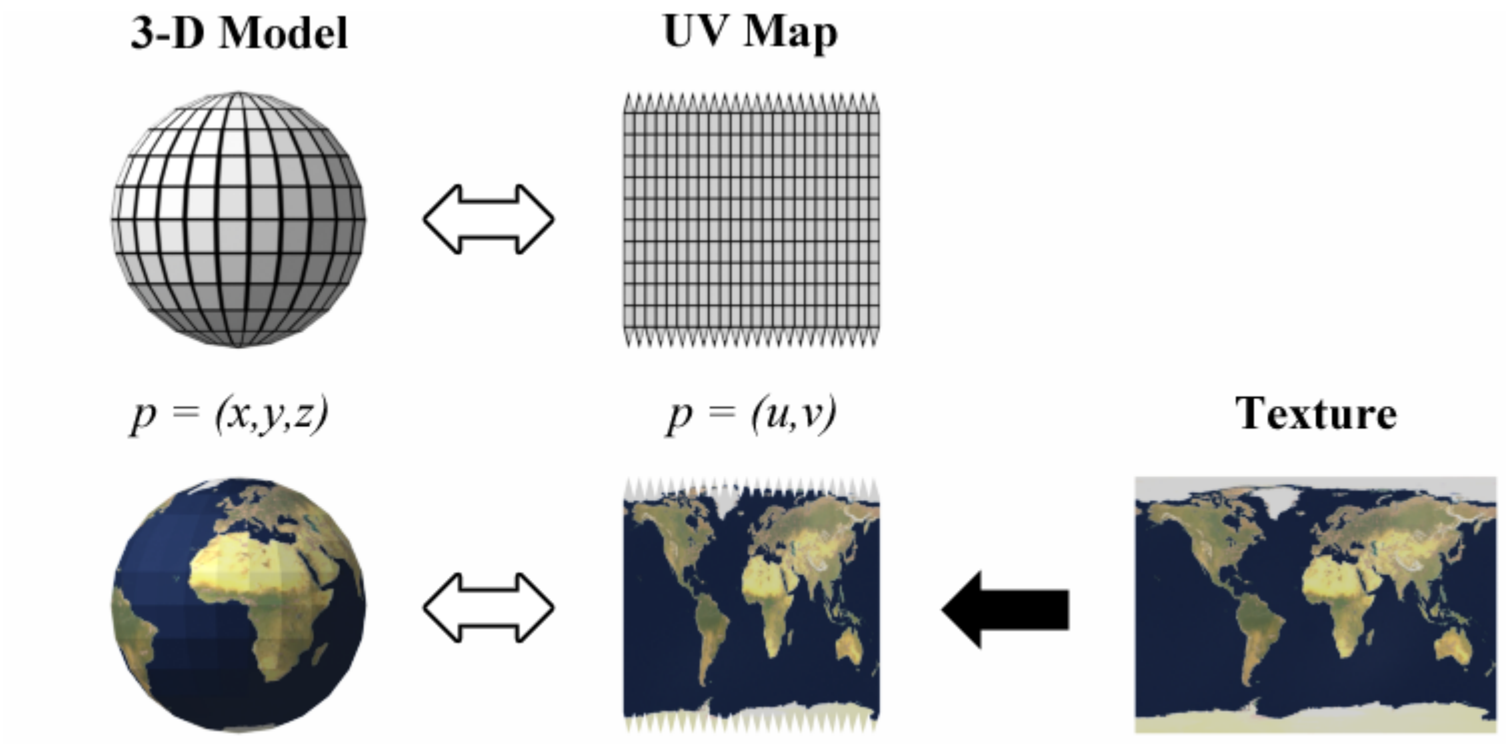
- Muzzle Flash
  - Added particles that emit as the spaceship shoots projectiles.
  - Once the spaceship stops shooting, there’s smoke that comes out from the gun of the spaceship.

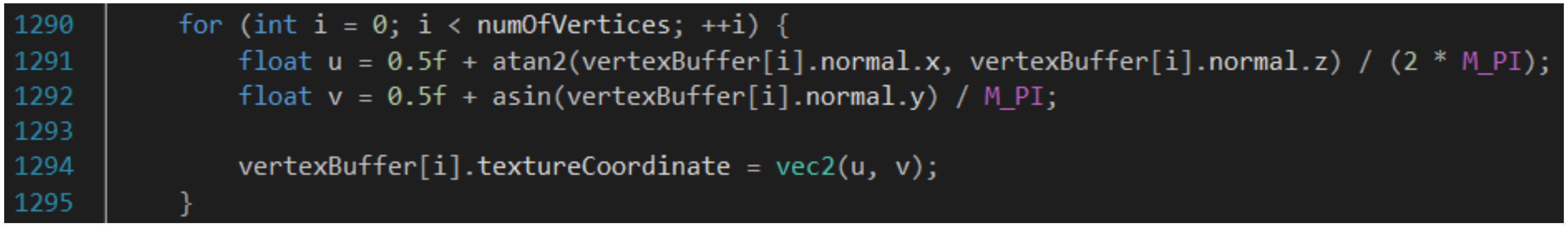
Miscellaneous Upgrades (Cont..)

- Implemented enemy spaceships
    - attached muzzle flash to them
    - integrated projectiles to enemy ships

- General scene upgrades
    - Added meteors into the scene
    - Gave them a velocity
    - Attached particles to them

Texture Mapping

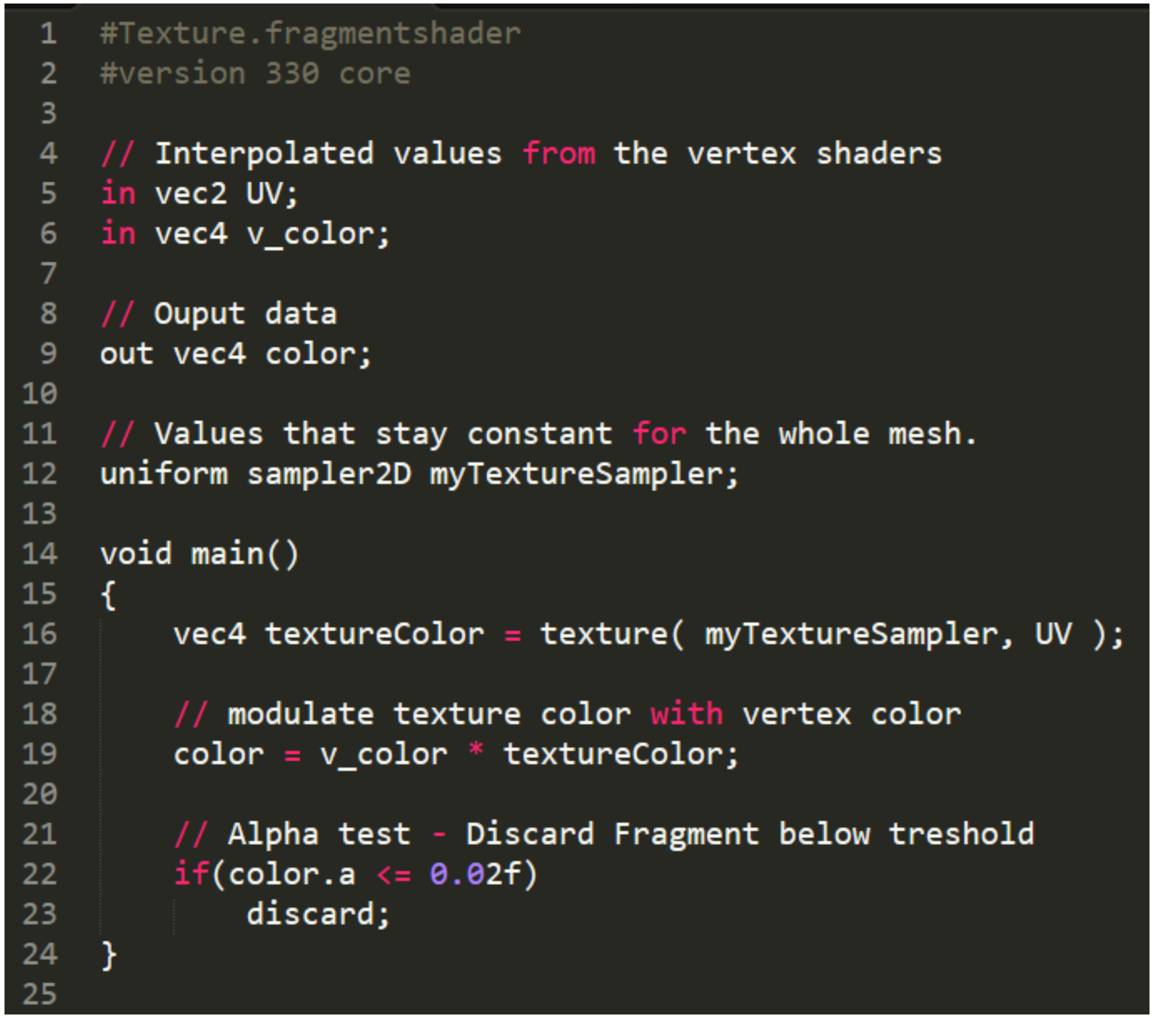




```
    for (int i = 0; i < numOfVertices; ++i) {
        float u = 0.5f + atan2(vertexBuffer[i].normal.x, vertexBuffer[i].normal.z) / (2 * M_PI);
        float v = 0.5f + asin(vertexBuffer[i].normal.y) / M_PI;

        vertexBuffer[i].textureCoordinate = vec2(u, v);
    }
```

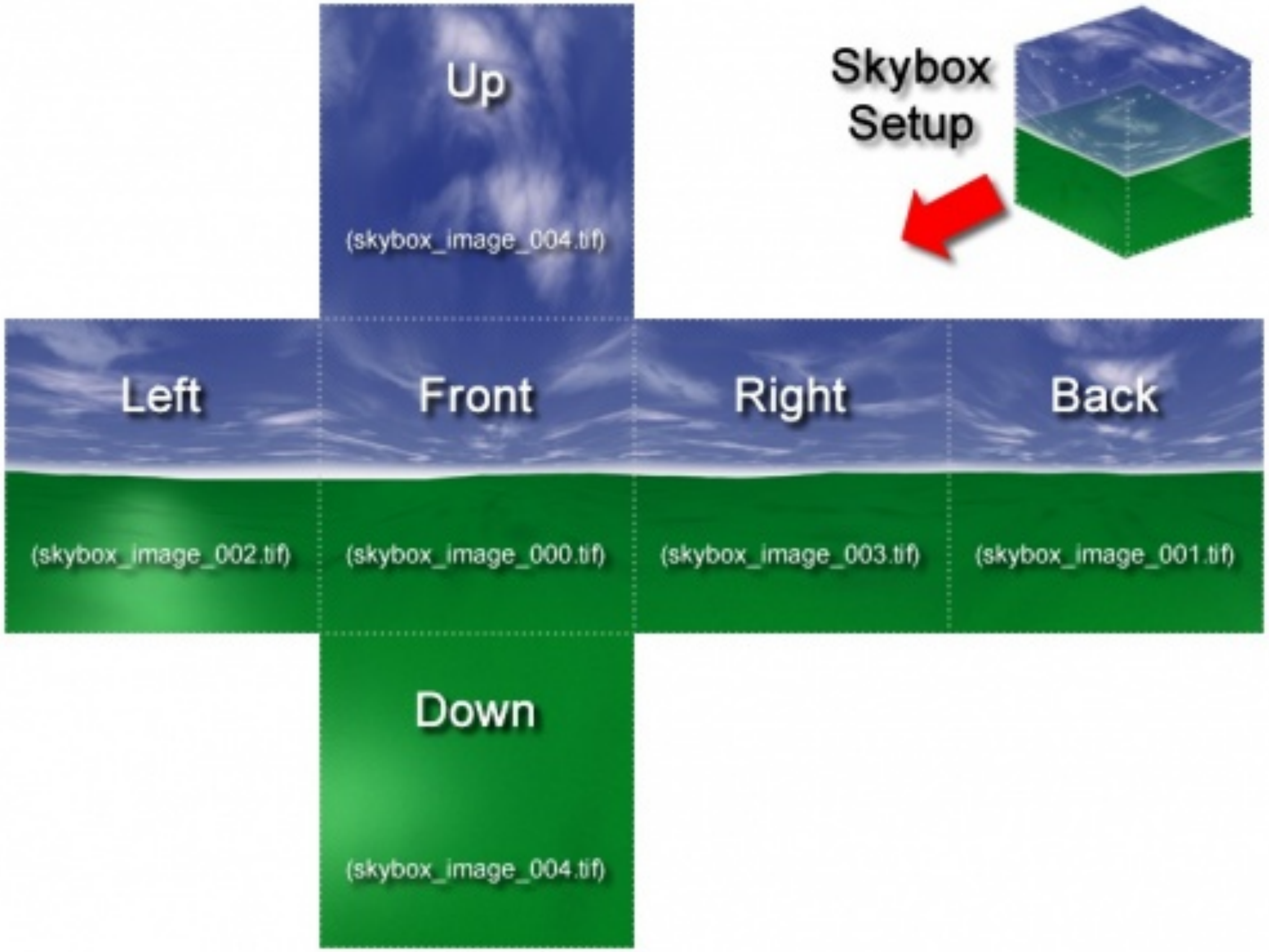
Texture Mapping Debugging

```
VertexBuffer[i]
.color
=
vec4(
    0.0f,
    0.0f,
    0.0f,
    1.0f
    )
```

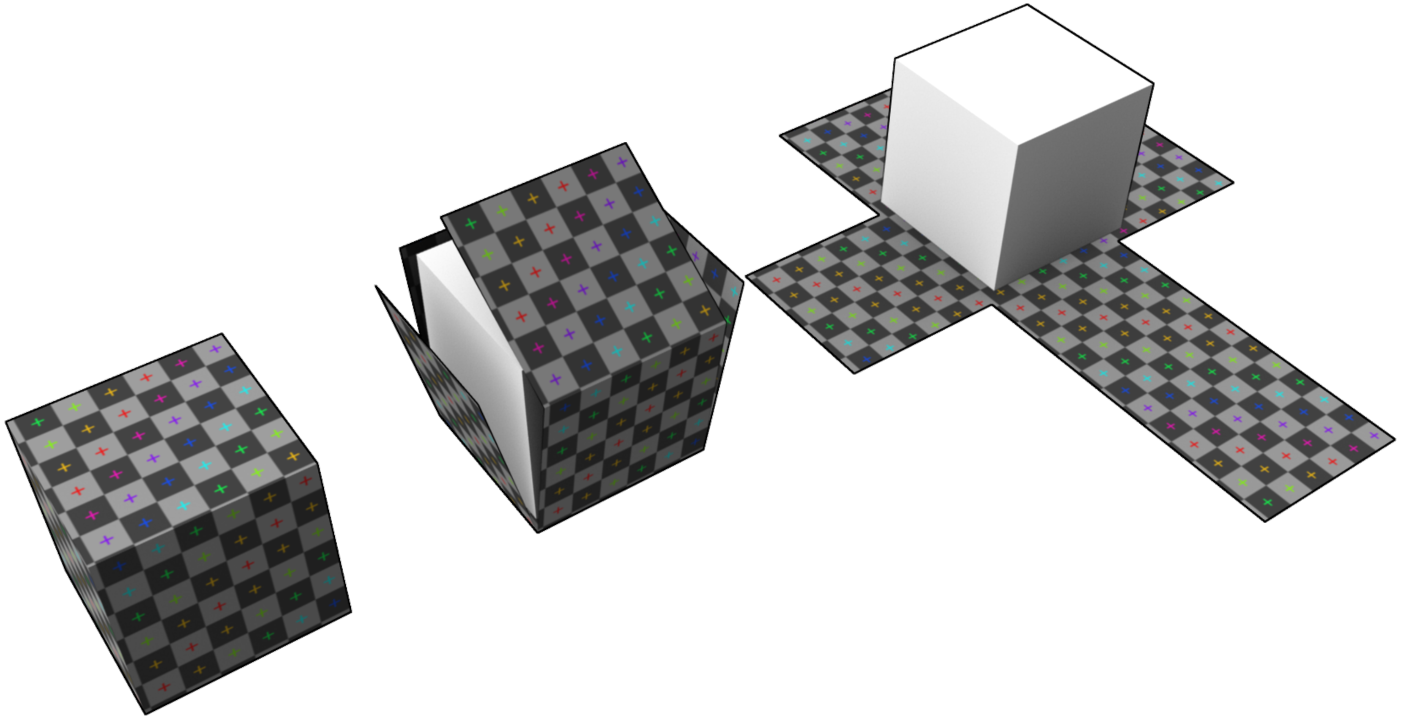


```
#Texture.fragmentshader
#version 330 core

// Interpolated values from the vertex shaders
in vec2 UV;
in vec4 v_color;

// Ouput data
out vec4 color;

// Values that stay constant for the whole mesh.
uniform sampler2D myTextureSampler;

void main()
{
    vec4 textureColor = texture( myTextureSampler, UV );

    // modulate texture color with vertex color
    color = v_color * textureColor;

    // Alpha test - Discard Fragment below treshold
    if(color.a <= 0.02f)
        discard;
}

```

Skybox


Up
(skybox_image_004.tif)
Skybox
Setup
Left
(skybox_image_002.tif)
Front
(skybox_image_000.tif)
Right
(skybox_image_003.tif)
Back
(skybox_image_001.tif)
Down
(skybox_image_004.tif)

Skybox



Spaceship & Projectiles

- Currently a stand alone object with a container for projectiles
- Will use similar logic as AsteroidSystem (based on ParticleSystem) due to efficiency

Thank You! :)