Discussion 10
CSE 167
Outline

- Tips on designing your simulation
  - With a scene graph
  - Without a scene graph
- Extra credit
  - Texture mapping
  - Sound effect
Reminder

- Project 4 due this Sunday @11:59 PM
- MAKE SURE TO READ THE INSTRUCTIONS CAREFULLY. ESPECIALLY THE SUBMISSIONS!
- Get ready for the finals!
  - All implementation so far in hw, but understanding the theory is important!
Real Time Simulations

- Soft Real-Time Systems
- Multiple agents (AI, moving objects in scene, player) interact with each other => change position, orientation, direction, etc
- In our case, two steps:
  - Update the scene
  - Render the scene
A Very High Level Implementation

- **In IdleCallback**
  - For each astronaut
    - Check for collisions with other collidable objects => update nonstatic objects, including current astronaut
    - Positions, orientation of astronaut changes => record them

- **Simple, right...?**
  - Usage of a scene graph complicates things...
  - Will show possible implementations with and without scene graph
Implementation Tips: No Scene Graph

- Keep two arrays
  - One for astronauts
  - Another for astronauts + static objects (boxes, walls)
- Check collisions between current astronaut and full list of other objects (do NOT check for self collisions)
  - Use collision detection logic described in past discussion section
ONLY IMPLEMENTATION IDEAS ARE COMING UP NEXT, NOT FULL IMPLEMENTATIONS

YOU DON’T HAVE TO FOLLOW THEM

FLEX YOUR DESIGN SKILLS IF YOU WANT
Some Pseudocode

- Again, just for inspiration

```cpp
class Mesh {
    public:
        // Ctor can deal with initializing tight bounding sphere
        Mesh(string path);

        void handleCollision(Mesh& other);  // Sphere-Sphere resolution
        void handleCollision(const Plane& plane);  // Sphere-Plane collision

        // Other methods like draw */
    
    private:
        BSphere sphere;
        // Other data members needed for rendering Mesh */
    };

    // Can also make it a struct
    class BSphere {
        public:
            BSphere(vec3 c, float r);
            float getRadius() const;
            vec3 getCenter() const;

        private:
            vec3 center;
            float radius;
    };

    class Plane {
        public:
            Plane(vec3 norm, vec3 pntOnPlane);
            float shortestDistToPnt(vec3 pnt);  // See Lecture #13 slides 8 - 12

        private:
            vec3 normal;
            float distance;
    };
```
HELP! MY SPHERES ARE STICKING TOGETHER

Just take your sphere and **PUSH IT SOMEWHERE ELSE**

From "HelixSnake's Top 50 Skate 3 Clips"
Push the Sphere Away: Sphere-Sphere

- Move sphere away from the other object so that they aren't touching
- If spheres are colliding, then subtract the distance between the centers from the distance needed for the spheres to touch and add a small epsilon (0.01f)
  - Use this and the surface normal of the collided sphere to translate current sphere away
  - Epsilon meant to keep spheres from touching
- Make sure to change orientation for other object if it’s supposed to move BEFORE you do this
Push the Sphere Away: Sphere-Plane

- Move sphere away from plane
- Follows same logic as sphere-sphere
Some Extra Tips

- Don’t spawn your astronauts in the exact same spot
  - Cannot push the sphere away in that case
- Modifying the speed of the astronaut using the time between frames can help if your machine is fast
  - deltaTime * speed * frontDir
- You can render a sphere as a representation of the bounding sphere
  - Can render it as a wireframe
So You’re Using A Scene Graph

- Example simple scene graph draws 3 astronauts
- Trick is to change rot, transl nodes of each astronaut
  - Rot\_i, transl\_i pair represent astronaut i
- Barebones
  - Up to you to decide how to stop scene graph from drawing dead astronauts (new node type?)

- Otherwise, collision handling/resolution is pretty much the same
Texture mapping

It is pretty similar to the skybox.

float texCoords[] = {
    0.0f, 0.0f, // lower-left corner
    1.0f, 0.0f, // lower-right corner
    0.5f, 1.0f  // top-center corner
};
Texture mapping

Reordering is important.

Create one more layer of VBO and pass in the texture coordinates.

```glsl
#version 330 core
out vec4 FragColor;

in vec3 ourColor;
in vec2 TexCoord;

uniform sampler2D ourTexture;

void main()
{
    FragColor = texture(ourTexture, TexCoord);
}
```
Sound effect

You are allowed to import any outside audio libraries.

Examples: OpenAL(3D sound), Irrklang(recommended), PortAudio(no idea about it)

So the pseudo code is like

```plaintext
if(players join || players leave){
  playSound("A");
}

if(player is moving ){
  playSound("B");
}
```
The Final Exam!

- 12/18/2020
- Quiz is open for 24 hour period, 3 hour time limit, should take 2 hours to complete
- Canvas quiz: Expect
  - To upload images so we can grade math problems
  - ...math problems for that matter (linear algebra, coordinate bases, etc)
  - Short answer questions (such as implementation, understanding of CG topics)
- Anything covered in lecture is FAIR GAME
Review Sessions/OH

- Review session during finals week
  - Vote in the Poll on Piazza for the Review Session time!
  - Check our office hours on Piazza. We will shift our hours around to accommodate last minute questions
Thank you and keep on rendering!