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

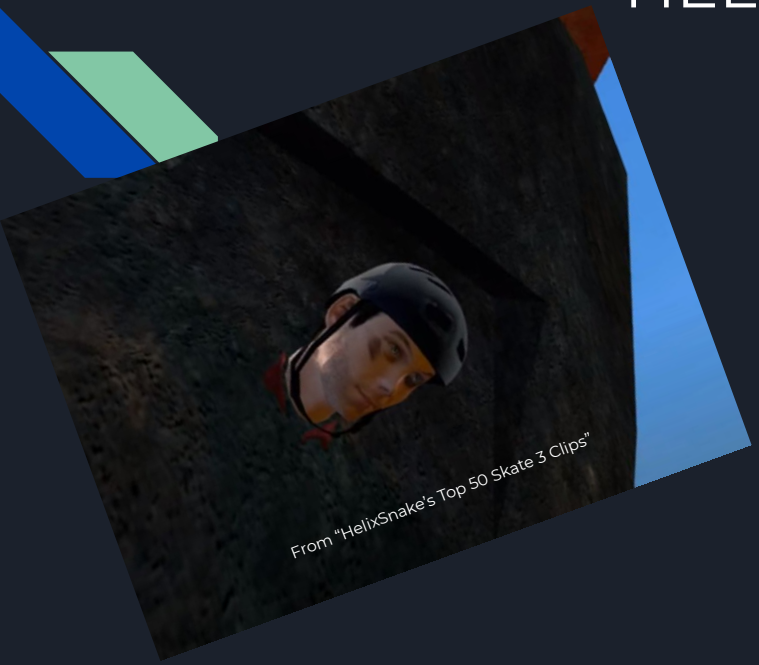
Some Pseudocode

- Again, just for inspiration

```
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;
};
```

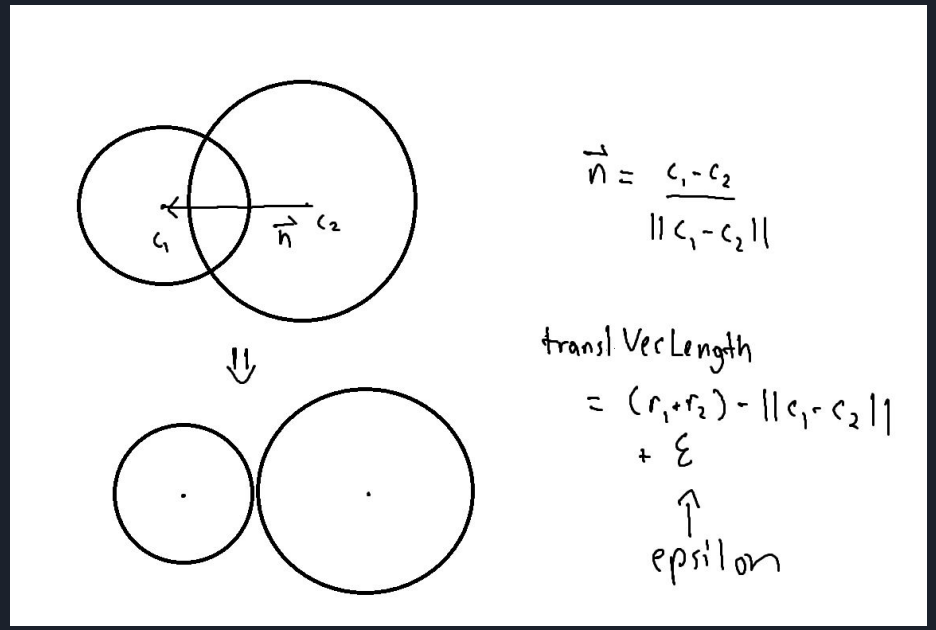
HELP! MY SPHERES ARE STICKING TOGETHER



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

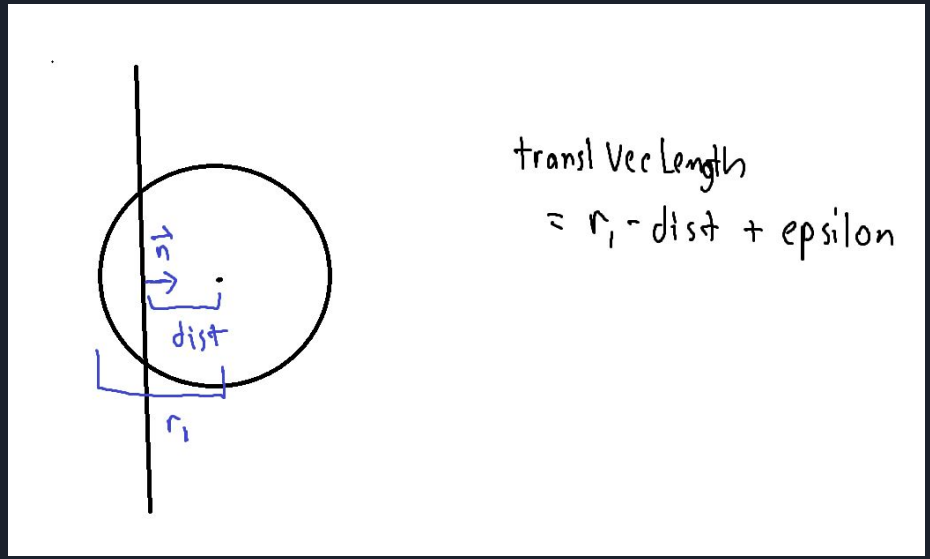
Push the Sphere Away: Sphere-Sphere

- 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



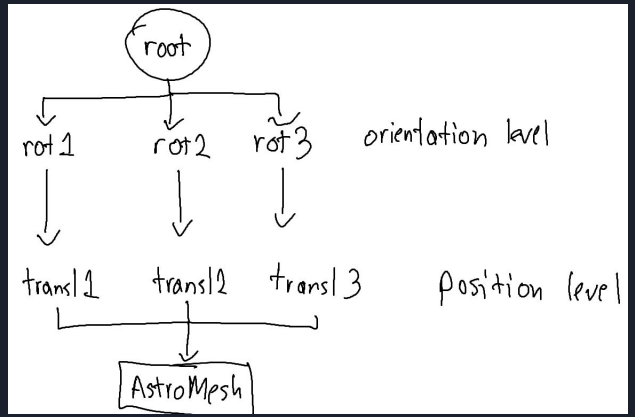
Some Extra Tips

- 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
    - https://stackoverflow.com/questions/137629/how-do-you-render-primitives-as-wireframes-in-opengl

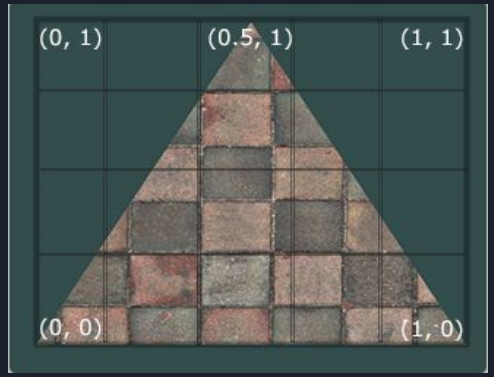
So You’re Using A Scene Graph

- 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?)

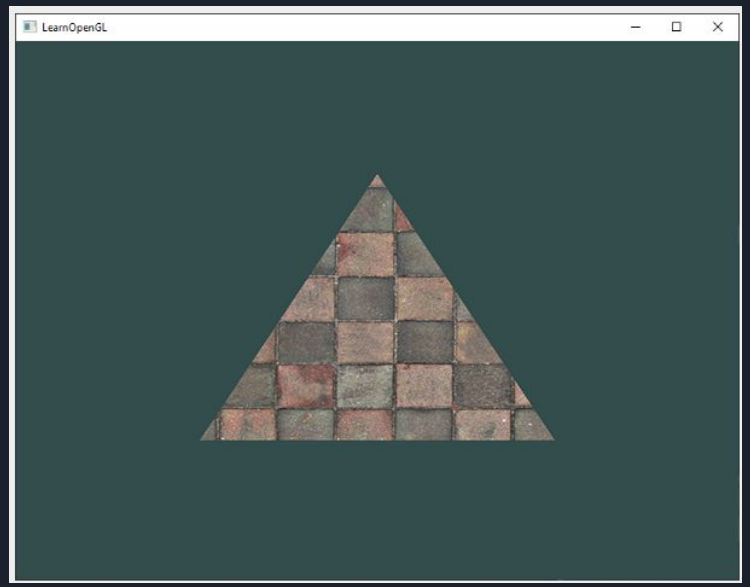
- Otherwise, collision handling/resolution is pretty much the same



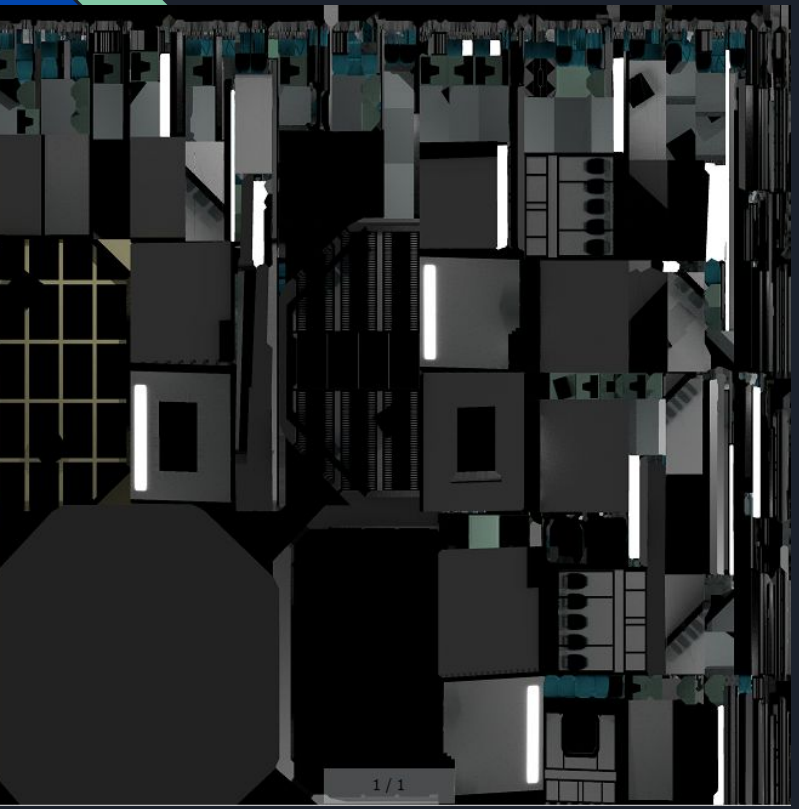
Texture mapping



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
};
```




```
vt 0.071115 0.949428
vt 0.070737 0.951234
vt 0.951384 0.384159
vt 0.070888 0.949682
vt 0.070549 0.951412
vt 0.951901 0.382112
vt 0.952080 0.381966
vt 0.070463 0.951508
vt 0.951787 0.385027
vt 0.951862 0.383615
vt 0.454277 0.950622
vt 0.454124 0.949171
vt 0.453976 0.948139
vt 0.453989 0.948128
vt 0.454290 0.950611
vt 0.454111 0.949182
vt 0.094571 0.950689
vt 0.094966 0.951268
vt 0.094785 0.951001
vt 0.094739 0.950139
vt 0.094563 0.949879
vt 0.951464 0.387482
vt 0.951810 0.387461
vt 0.951645 0.387655
vt 0.951819 0.386885
vt 0.952118 0.386867
vt 0.952006 0.387135
vt 0.951624 0.387211
vt 0.951255 0.386954
vt 0.951626 0.386350
vt 0.951985 0.386614
vt 0.951425 0.386676
vt 0.951895 0.385931
vt 0.952174 0.386863
vt 0.952067 0.386485
```

```
f 5/7/3 8/8/3 6/9/3
f 7/10/4 1/3/4 8/11/4
f 7/12/5 3/13/5 2/14/5
f 1/15/6 6/16/6 8/17/6
f 10/18/7 12/19/8 9/20/8
f 11/21/6 14/22/6 12/23/6
f 13/24/9 16/25/10 14/26/10
f 16/27/5 10/28/5 9/29/5
f 15/30/11 11/31/11 10/32/11
f 9/33/12 14/34/12 16/35/12
f 18/36/7 20/37/7 17/38/7
f 20/39/6 21/40/6 22/41/6
f 21/42/10 24/43/10 22/44/10
f 24/45/5 18/46/5 17/47/5
f 23/48/13 19/49/13 18/50/13
f 17/51/12 22/52/12 24/53/12
f 26/54/8 28/55/7 25/56/7
f 27/57/11 30/58/11 28/59/11
f 29/60/9 32/61/10 30/62/10
f 31/63/12 25/64/14 32/65/14
f 31/66/5 27/67/5 26/68/5
f 25/69/6 30/70/6 32/71/6
f 34/72/7 36/73/8 33/74/8
f 35/75/11 38/76/11 36/77/11
f 37/78/10 40/79/10 38/80/10
f 39/81/14 33/82/12 40/83/12
f 34/84/5 37/85/5 35/86/5
f 33/87/6 38/88/6 40/89/6
f 42/90/8 44/91/8 41/92/8
f 43/93/13 46/94/13 44/95/13
f 45/96/10 48/97/10 46/98/10
f 47/99/14 41/100/12 48/101/12
f 42/102/5 45/103/5 43/104/5
f 41/105/6 46/106/6 48/107/6
f 50/108/7 52/109/8 49/110/8
```

Texture mapping

Reordering is important.

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

```
#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

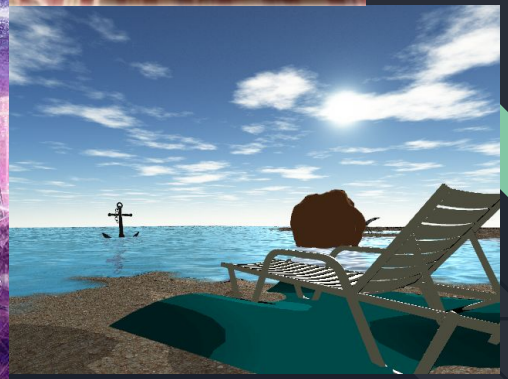
```
if(players join  ||  players leave){

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!