Outline

- Movement and Collision Resolution
- Particle Systems
Astronauts

- Face direction they walk in
- No animation is required
- Maximum of 10 (players/non-players)
- Player is controllable
- Non-players randomly appear and disappear from lobby
Movement - Player

- Use keyboard callback like we did for previous PAs
  - WASD/Cursor keys to control one astronaut
  - Because we have camera rotations, keep each direction constant
    - W/Up - North
    - A/Left - West
    - S/Down - South
    - D/Right - East

- if (collision)
  - Reference discussion 8 slides for collision detection
Environment Collision

- **Bounding Planes**
  - 4 axis aligned walls
  - 2 diagonal walls

- **Bounding Spheres**
  - 2 boxes

- **You can find these through trial and error**
  - Track position as you move a sphere around to estimate
Movement - Player (Collision Resolution)

- Sphere-Plane/Sphere-Sphere collision
  - Player collides with environment/other players
  - Loop through objects and detect for collision
    - Upon collision, stop player movement until collision is resolved or new direction is given
    - You can attempt to have the wall pushback the player, but it is not required
Movement - Non-Player

- Spawns in at random time intervals
  - Starts walking in random direction after randomized period of time
- Randomly stops moving for a bit before they start walking again
Movement - Non-Player (Collision Resolution)

- Angle of incidence = angle of reflection
- Sphere off of plane
  - Normal is normal of plane
Movement - Non-Player (Collision Resolution)

- Sphere off of sphere
  - Normal is vector between centers
- No need to account for total velocity
  - Assume same velocity/mass for all non-players
Collision interactions

- Treat non-moving non-players as static objects
- Only moving non-players will bounce like previous slide when collision occurs
  - This means if a moving non-player collides with a non-moving non-player, the non-moving non-player should not start moving
- Player should remain separate, does not bounce upon collisions
  - Player should only follow keyboard controls, player should stop moving upon collision
Tips

- Use rand() for a random integer between 0 and RAND_MAX
  - how to generate random float in range?

- glfwGetTime() to get seconds since the window was initialized
  - how to get delta time?

- Use your scene graph from PA3, but keep it simple
Particle System

- Particle effect when astronauts appear and disappear
- Needs to last ~3 seconds with at least 100 particles
- Two parts:
  - Simulation
    - Spawn particles if necessary, check lifetime
    - Move particles
  - Rendering
    - Update positions in VBO
    - Draw
Particle System - Header

- This is just an example header file add/modify methods and member variables as you see fit
- Some things you might want:
  - Particle struct which stores single particle data (e.g., position, velocity, color, life)
  - Separate data structures for full particle data used for simulation, and particle data only passed to the shader
Particle System - Simulation

- Call an update() to the particle system in Window::idleCallback() to update the particle system every frame
- Increment a life counter in each particle to keep track of how long a particle has existed
- Consider using delta time to keep particle movement consistent over varying fps
  - i.e., position += delta_time * velocity
You have many options for rendering particles. Feel free to explore!

**GL_POINTS**
- You’ve already done this in PA1
- Easy to do, but least flexible appearance-wise

**Point Sprites**
- Small extension onto GL_POINTS to render textures onto points

**Instanced Rendering**
- Allows the use of arbitrary geometries for particles
- Also used to manually implement billboarded sprites in a similar fashion to Point Sprites.
Particle System - Useful Methods

- **glBufferSubData**
  - Similar to glBufferData, but does not allocate memory. Is more efficient if you want to replace data inside your VBO.
  - Use after updating particles.
  - Remember to bind a VBO before using!

- **rand**
  - Add some variation in particle initialization and/or movement.

- **glDrawArraysInstanced/glDrawElementsInstanced** and **glVertexAttribDivisor** (optional if using Instanced Rendering)
  - Enables the rendering of the same object many times.
Any Questions?