Assignment 4 Overview
Parsing Joints and Weights

Read in PMD data via MMDAdapter class: (getJoint() and getJointWeights())

Useful structs (included):

• Bone (bone_geometry)
• Skeleton (bone_geometry)
• SparseTuple for weights (mmdadapter)

May want to create:

• Joint struct
• LineMesh struct (for bone visualization)
Visualizing Bones

Create LineMeshes for rendering in procedure_geometry

LineMesh can represent bones (lines), pickable bones (cylinders), and bone axes (lines)

Cylinders represented by two types of lattices:
- Line segments following the length of the bone
- “Circles” (really just n-gons) at the bone’s end and middle positions
Bone Picking

How to select an object in 3D (world) space based on the mouse position in 2D (screen) space?
Ray Casting

Generate a ray from screen space coordinates to world space (ScreenToWorld)

How to do this?
Camera Matrix

• Compute the NDC (normalized device coordinates) from the screen position
• Determine coordinates in world space based on camera position and orientation
GLM::Unproject

Same idea as reversing the camera matrix, but handled by glm!

Must create a ray based on this unprojected position in world space
  • Subtract eye position
  • Unproject two positions along z to find ray
Cylinder Intersection

Intersection must be within cylinder radius and height

Intersect with cylinder in one of three ways:

1. Intersect based on cylinder’s position in world coordinates (inefficient)

2. Intersect based on cylinder’s position in bone’s local coordinates (transform ray into this coordinate system)

3. Intersect based on 2D projection of cylinder (check if hit within radius then compute intersect points to check height)
Object Picking Using Pixel Color

Kind of a hack!

- Create a buffer (not seen by viewer) that renders all objects as different colors
- Map user’s screen selection to this buffer and check pixel’s color

http://www.lighthouse3d.com/tutorials/opengl-selection-tutorial/
Bone Manipulation

GUI class handles keyboard and mouse callbacks

 Mostly similar to what you did in Menger…

 …except you’ll also need to update the bone in the deformed coordinate system (will want some additional handling in bone_geometry)