Discussion 7
CSE 167

Outline

- Disco Ball
- Scene Graph Example
- Texture Coordinate Parsing
- Camera Control
- Extra Credit
  - 3D Modeling
  - Dynamic Environment Mapping

Disco Ball

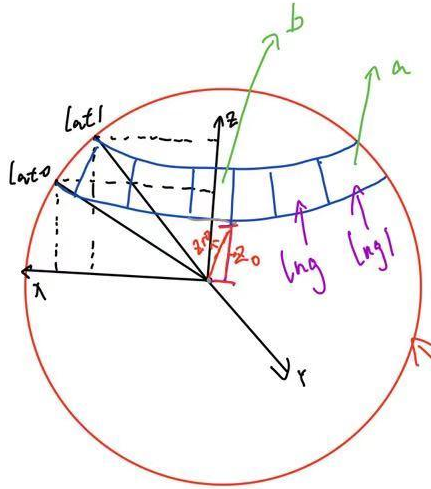
- Mirror reflection effect with low polygon ball model
- Create polygon mesh for ball with adjustable number of quads
- Add environment mapping to shader files shader.vert and shader.frag
- Lighting code is no longer required here
- Tutorial link: https://learnopengl.com/Advanced-OpenGL/Cubemaps

Disco Ball

https://github.com/cbutarbu/Sphere_167

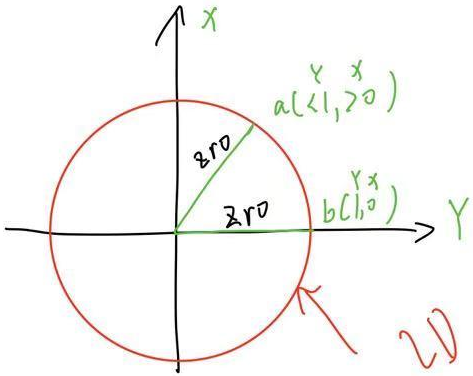
If you want the reflection to look better, make stackCount and sectorCount in Sphere.h a larger number.

If the environment mapping is upside down, make sure you follow the reflection part instead of refraction part in the tutorial.



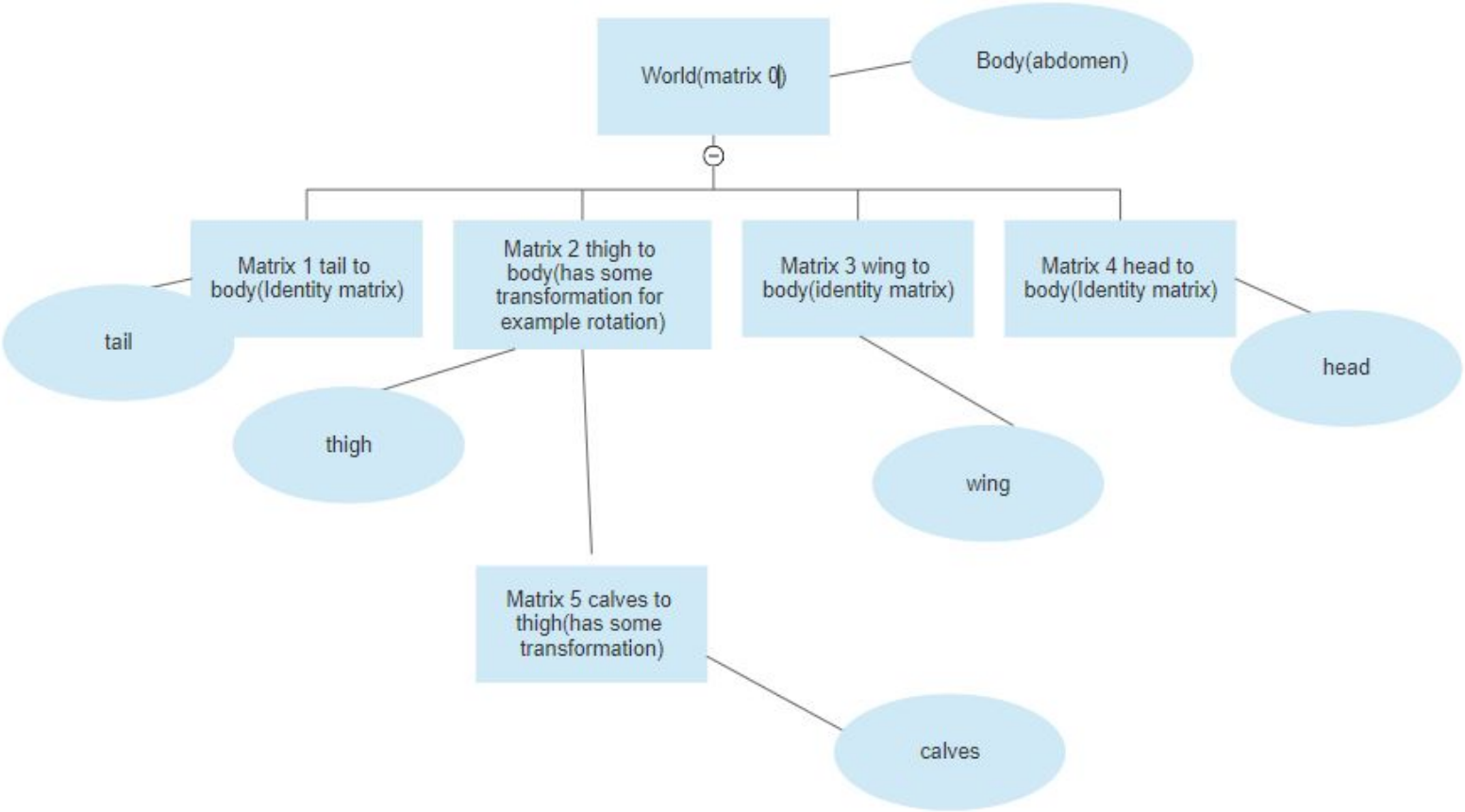
For Sphere.cpp

3D sphere



2D circle

Scene Graph Example


World(matrix 0)
Body(abdomen)
Matrix 1 tail to body(Identity matrix)
Matrix 2 thigh to body(has some transformation for example rotation)
Matrix 3 wing to body(identity matrix)
Matrix 4 head to body(Identity matrix)
tail
thigh
wing
head
Matrix 5 calves to thigh(has some transformation)
calves

Refer their relations from last slide. Mat2->draw(M0);

{

thigh->draw(M0*M2);

Mat5->draw(M0*M2) {calves->draw(M0*M2*M5);}

}

That is just an example. You should do something like

vector<Object> obj.push_back(head…..thigh); obj[i]->draw(.....);

Please refer Discussion 6 for more details.

Loading Models with Texture Coordinates

- Modify your obj parser to also parse ‘vt’ lines, as well as the texture coordinate indices in the ‘f’ lines
- Add VBO for texture coordinates (vec2)
- Note: Indices may not necessarily match as assumed in PA2. Reordering vertex data may be necessary

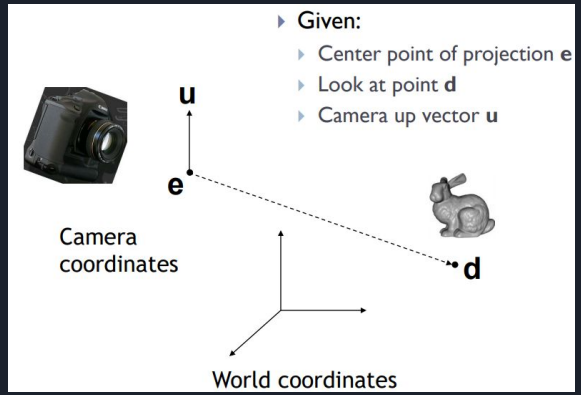
```
for (unsigned i = 0; i < vertex_indices_.size(); i++) {
  vertices_.push_back(input_vertices[vertex_indices_[i]]);
  uvs_.push_back(input_uvs[normal_indices_[i]]);
  normals_.push_back(input_normals[normal_indices_[i]]);
  indices_.push_back(i);
}
```

```
f 1/1/1 5/2/1 7/3/1 3/4/1
f 4/5/2 3/6/2 7/7/2 8/8/2
f 8/8/3 7/7/3 5/9/3 6/10/3
f 6/10/4 2/11/4 4/12/4 8/13/4
f 2/14/5 1/15/5 3/16/5 4/17/5
f 6/18/6 5/19/6 1/20/6 2/11/6
```

- ‘f’ lines with 4 elements indicate GL_QUAD usage. You can reinterpret this into two triangles, or triangulate the obj file using software like Blender.

Camera Control

- User needs to be able to move forward, back, up and down, as well and turn the camera left or right
- Change Window::eyePos to move
- Change Window::lookAtPoint to rotate
- Keep Window::upVector constant
- Remember to recompute Window::view when parameters change




```
18  // View Matrix:
19  glm::vec3 Window::eyePos(0, 0, 20);                    // Camera position.
20  glm::vec3 Window::lookAtPoint(0, 0, 0);          // The point we are looking at.
21  glm::vec3 Window::upVector(0, 1, 0);             // The up direction of the camera.
22  glm::mat4 Window::view = glm::lookAt(Window::eyePos, Window::lookAtPoint, Window::upVector);
```

Camera Control - Tips

- To keep camera facing in same direction while moving, update lookAtPoint as you update eyePos
- Recall how to rotate a point around another point ($T^{-1}RT$) in order to rotate lookAtPoint around eyePos
- Try to reuse your trackball code from PA2 to acquire the axis and angle required to rotate lookAtPoint around eyePos (not required)

Extra Credit

- Rider View
  - Add camera node to scene graph
- Custom sky box
  - Tutorial Here
    - Linked app may not be downloadable. Google Street View also takes 360 pictures
- Custom Objects
- Dynamic Environment Mapping

Creating 3D models

- Create a textured non-trivial object
- Use a 3D modeling software, like Blender or Maya
- Blender tutorials
  - Basic Modeling
  - UV mapping
    - don't worry about nodes and shaders, focus on uv mapping
- Export to .obj

Dynamic Environment Mapping

- Place camera in middle of disco ball with square dimensions and fov set to 90 and create renders in all 6 directions to create a cube map
- Make use of Frame Buffer Objects (FBO)
  - use the learnOpenGL mirror code as a reference