Programming Assignment 3

Submission

Deadline: 23:59:59, Sunday, May 12th , 2019 (KST, +0900)

- Github server clock

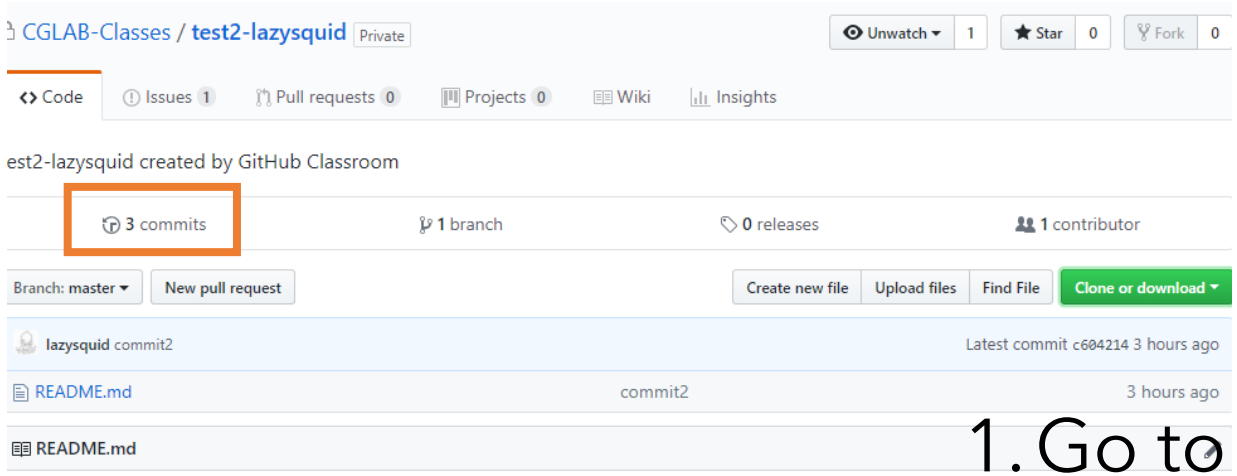
To submit your assignment, you must do two things. **Both of them must be done BEFORE deadline.**

1. You should push your commit to your assignment repo before deadline.
2. You should comment the last commit (before deadline) id (SHA-1 hash) in github issue board. (See next slide)

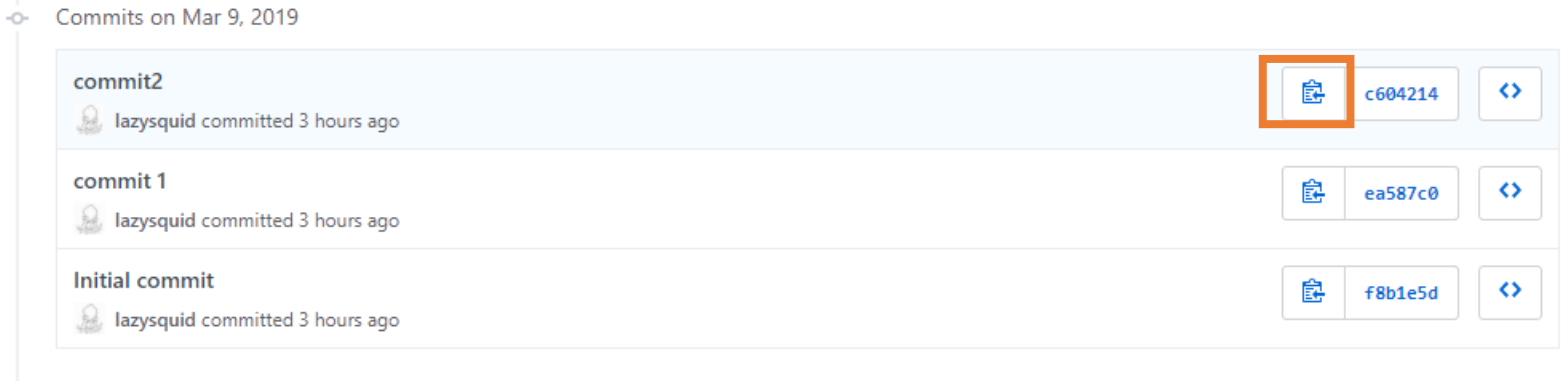
The last commit BEFORE dead line will be considered as submitted assignment.

- Github server will track this for me.
- Timestamp in your commit (local time) will be igrnoed. (I will use github server timestamp instead)

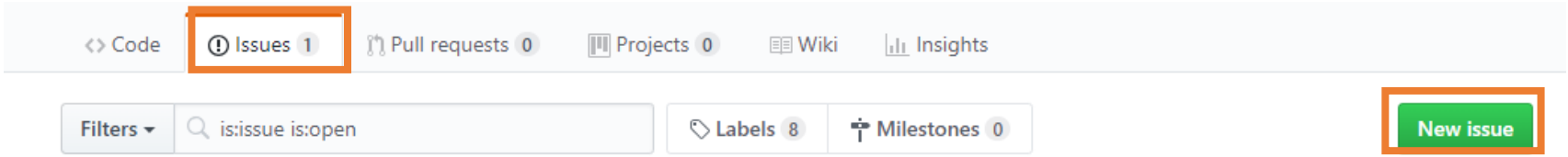
Commenting Commit ID 1/2

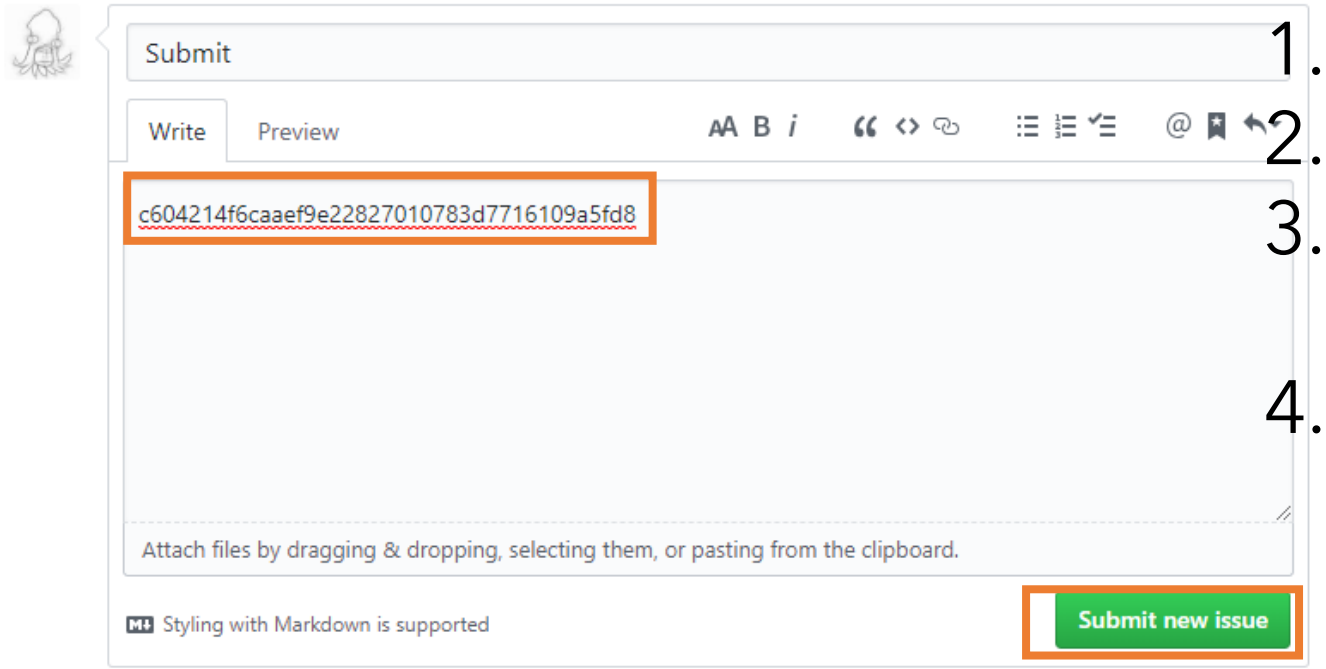


1. Go to your assignment repository
2. Click commits
3. Click copy button of your last commit



Commenting Commit ID 2/2





1. Go to issue tab
2. Click “new issue”
3. Paste your lastest commit id (Ctrl-v)
4. Click “submit new isse”

Policy

In the following cases, your grade for this PA will be 0

- Late submission (Late push before deadline or Late last commit id comment on issue board)
- Build/execuition failure
- Making public of your assignment repository
- If you tried to push your commit with force option(Tried to change history of remote server)

Your final grade will be “F”

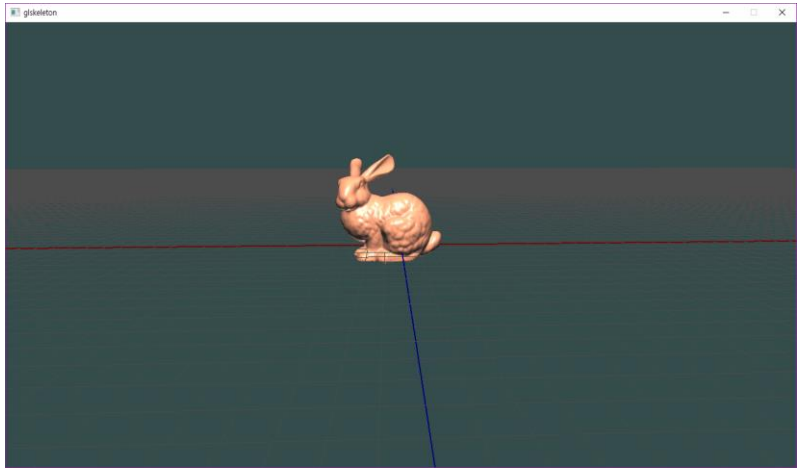
- Copy

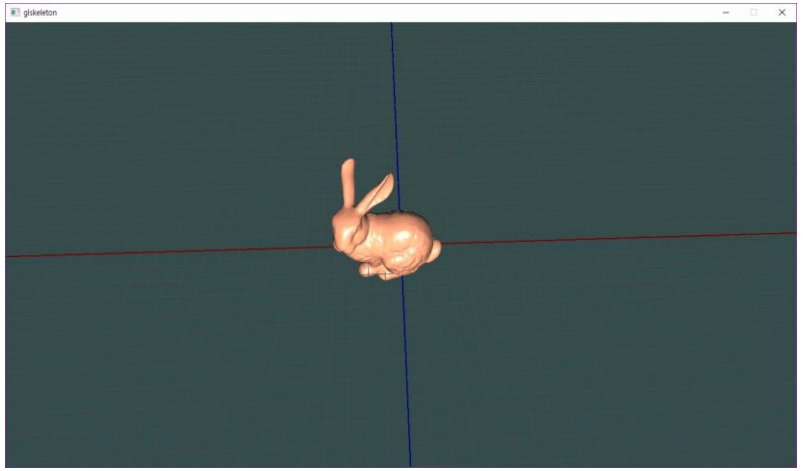
Task Lists

1. Implement trackball camera [18 Points]
2. Lighting [10 Points]
3. Report [2 Points]
    - Write your name, student id, github id in report.md [1 Points]
    - Attatch at least two result images in report.md [1 Points]

Task – Trackball Camera

- **Fix the lookat position to world origin**
- Rotate your camera by **dragging** [12 Points]
- Dolly in and dolly out by **scrolling** [3 Points]
  - Move your camera toward/backward to lookat direction.
- Zoom in and zoom out [3 Points]
  - Use key callback to do this. “q” for zoom in, “w” for zoom out

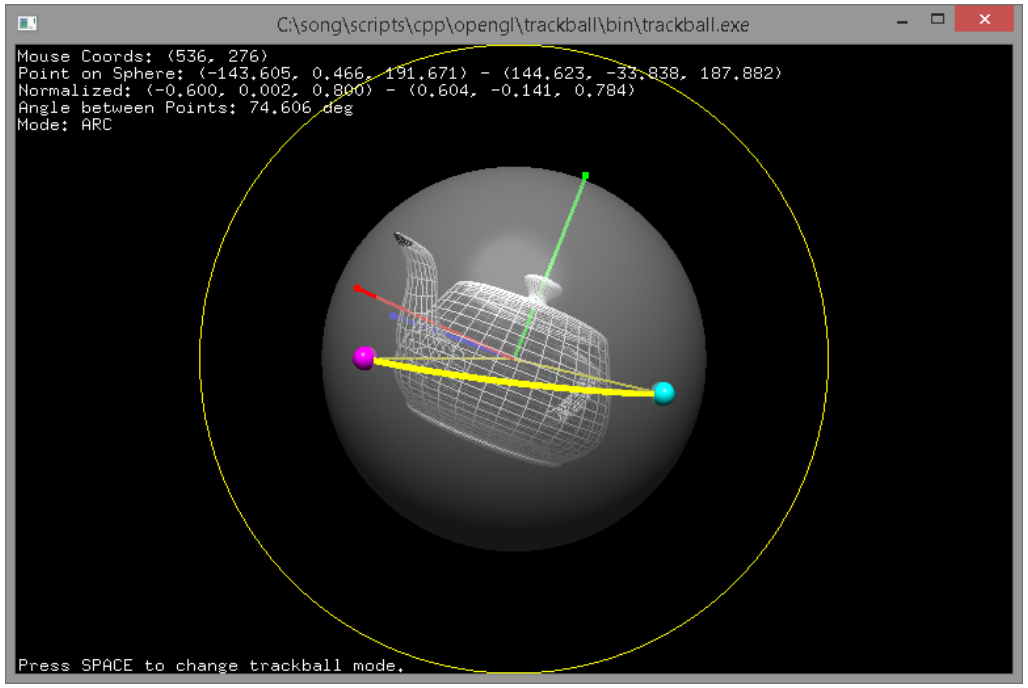




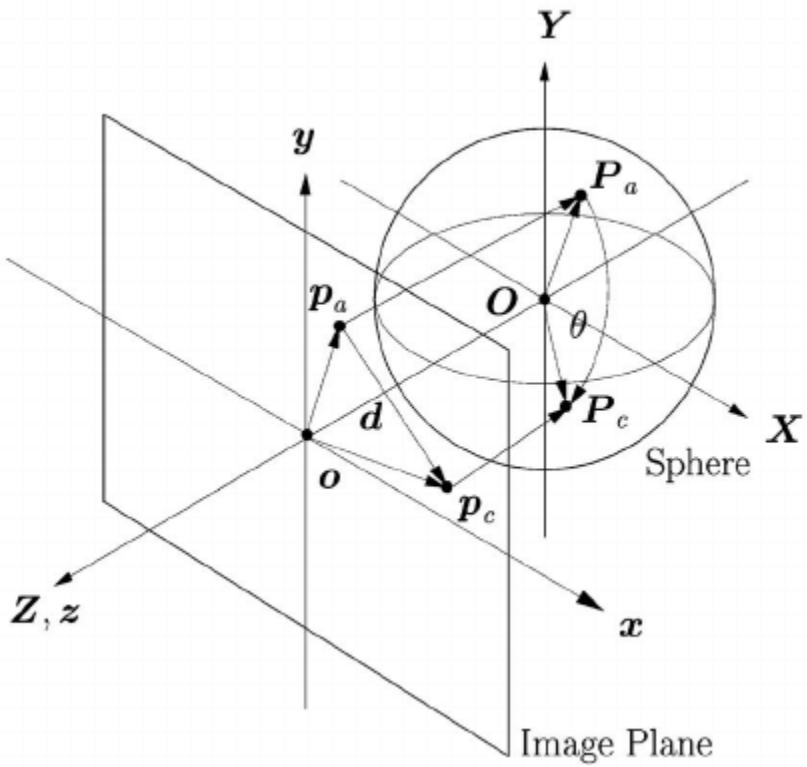
Task - Lighting

- You should apply phong shading model(ambient/diffuse/specular) and gurad shading option.
- Set one point light [5 Point] with on/off functionality.
  - ON, key “1”
  - OFF key “2”
- Set one directional light [5 Point] with on/off functionality.
  - ON, key “3”
  - OFF key “4”

Trackball Camera



http://www.songho.ca/opengl/gl_camera.html



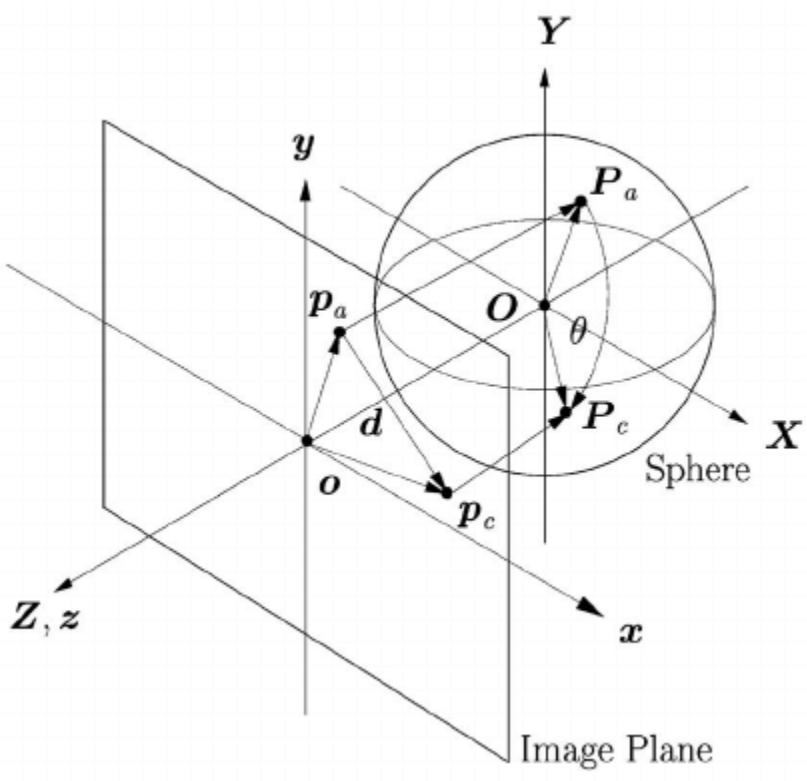
https://www.researchgate.net/figure/A-virtual-trackball-can-be-thought-of-as-a-3D-sphere-located-behind-the-screen-The_fig2_8329656

The camera which is orbiting around virtual sphere.

Trackball Camera

How to

1. Project your cursors on the sphere
2. Compute rotation matrix with two vectors. (Projected cursor of current and previous frame)
3. Apply the rotation matrix to your camera
   - You need to figure out how to do this.



Rotation Between Two Vector

```
#include <glm/gtx/quaternion.hpp>
// reference
// http://www.opengl-tutorial.org/kr/intermediate-tutorials/tutorial-17-quaternions/

glm::quat RotationBetweenVectors(glm::vec3 start, glm::vec3 dest) {
  start = glm::normalize(start);
  dest = glm::normalize(dest);

  float cosTheta = dot(start, dest);
  glm::vec3 rotationAxis;

  if (cosTheta < -1 + 0.001f) {
    // special case when vectors in opposite directions:
    // there is no "ideal" rotation axis
    // So guess one; any will do as long as it's perpendicular to start
    rotationAxis = cross(glm::vec3(0.0f, 0.0f, 1.0f), start);
    if (glm::length2(rotationAxis) <
        0.01) // bad luck, they were parallel, try again!
      rotationAxis = cross(glm::vec3(1.0f, 0.0f, 0.0f), start);

    rotationAxis = normalize(rotationAxis);
    return glm::angleAxis(glm::radians(180.0f), rotationAxis);
  }

  rotationAxis = cross(start, dest);

  float s = sqrt((1 + cosTheta) * 2);
  float invs = 1 / s;

  return glm::quat(s * 0.5f,
                   rotationAxis.x * invs,
                   rotationAxis.y * invs,
                   rotationAxis.z * invs);
}
```

Rotation is usually represented with quaternion. Understanding quaternion is out of scope.

But we can convert them into rotation matrix using glm.

So use this like this

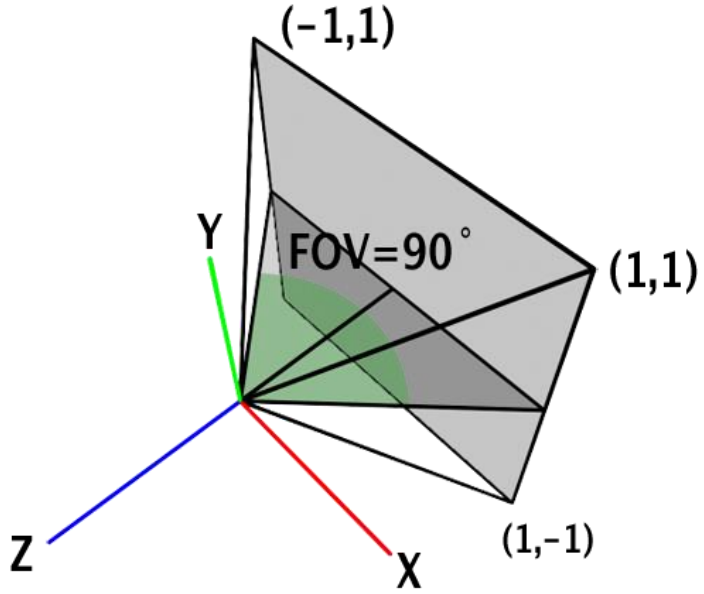
```
const glm::mat4 R =
        glm::toMat4(RotationBetweenVectors(v1,v2));
```

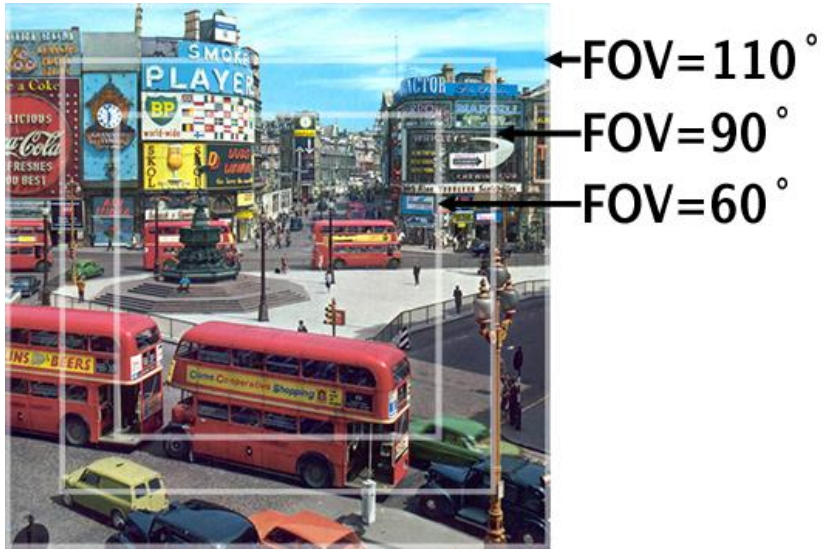
Dolly In/Out

- Translation of camera origin
  - Move your camera toward/backward to lookat direction.
- It affects to camera(view) matrix

Zoom In/Out

- Changing fov of camera
- It affects to projection matrix





PA3 Link

1. Login to github
2. Go to following link https://classroom.github.com/a/Knkv-pCu
3. Accept the assignment