Homework 1

COSE331 Computer Graphics

Goal

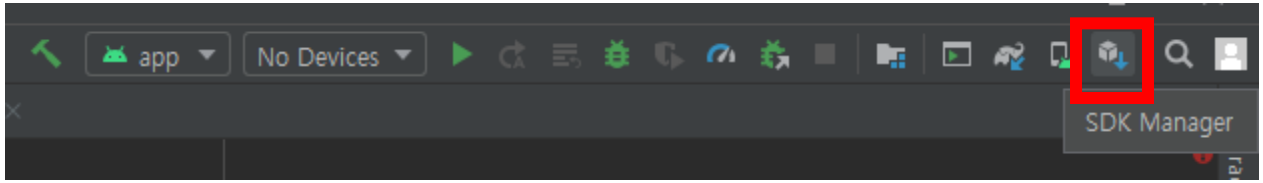
- Setting up Android Studio development environment
- Filling in some lines in the vertex shader
  - Vertex position
  - Vertex normal
  - Texture coordinates
- Filling in lines in the functions of the scene class in GL program

Android Studio

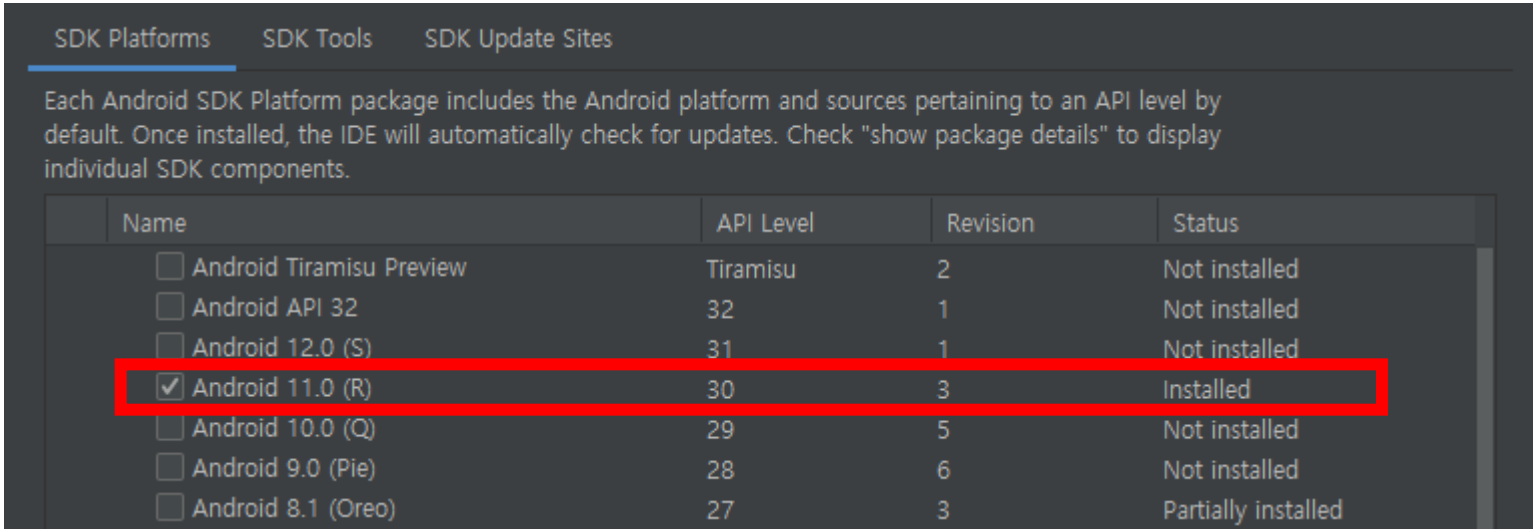
- **Android Studio** is the official integrated development environment (IDE) for the Android platform.
- Android Studio can be downloaded from the official website. [link]

Android SDK

- **Android SDK** can be installed through the SDK Manager in Android Studio.



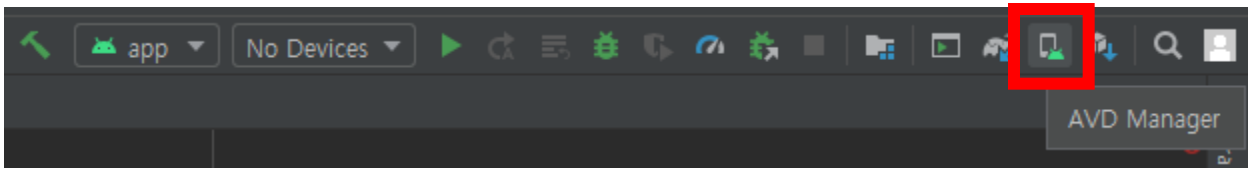
- Use SDK **API level 30**



Android SDK

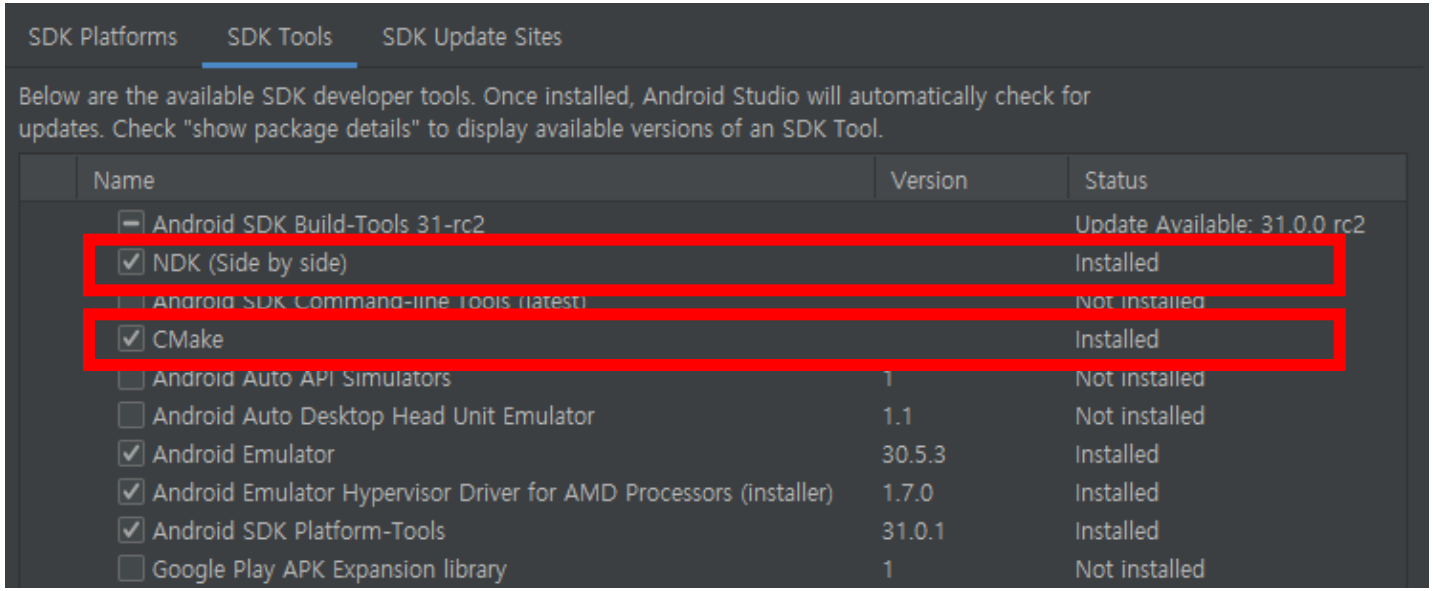
- You may need to install **additional platforms** to try USB debugging on your smartphone.

- Or you can use **Android Virtual Device (AVD)**.
  - To use AVD, you need to enable **hardware virtualization technology** (shown as **VT-x** or **SVM**) in BIOS settings.



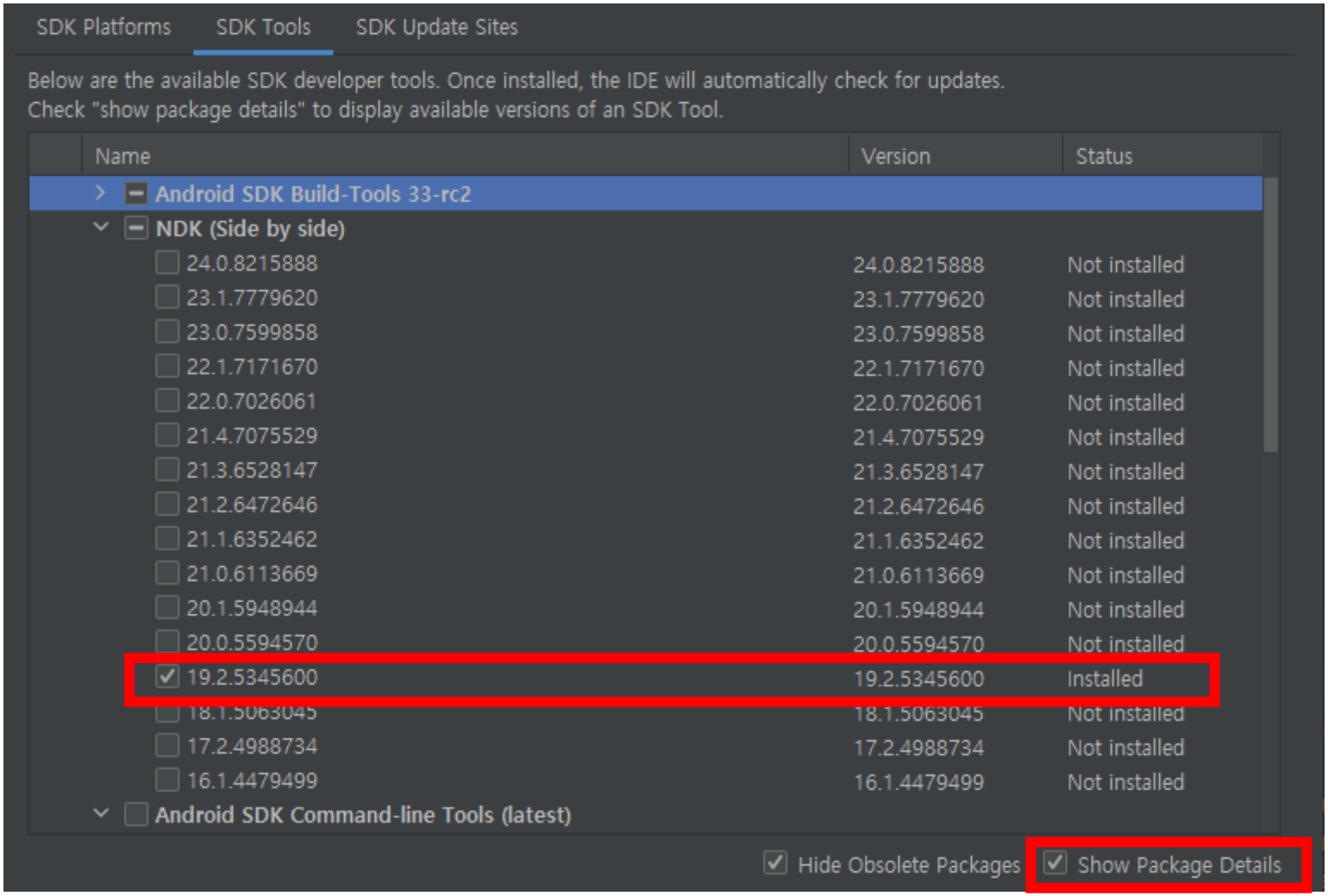
Android SDK

- To use **C++ native language** on Android, you need to install the following tools.
  - CMake
  - NDK **(Use version 19.2.5345600)**



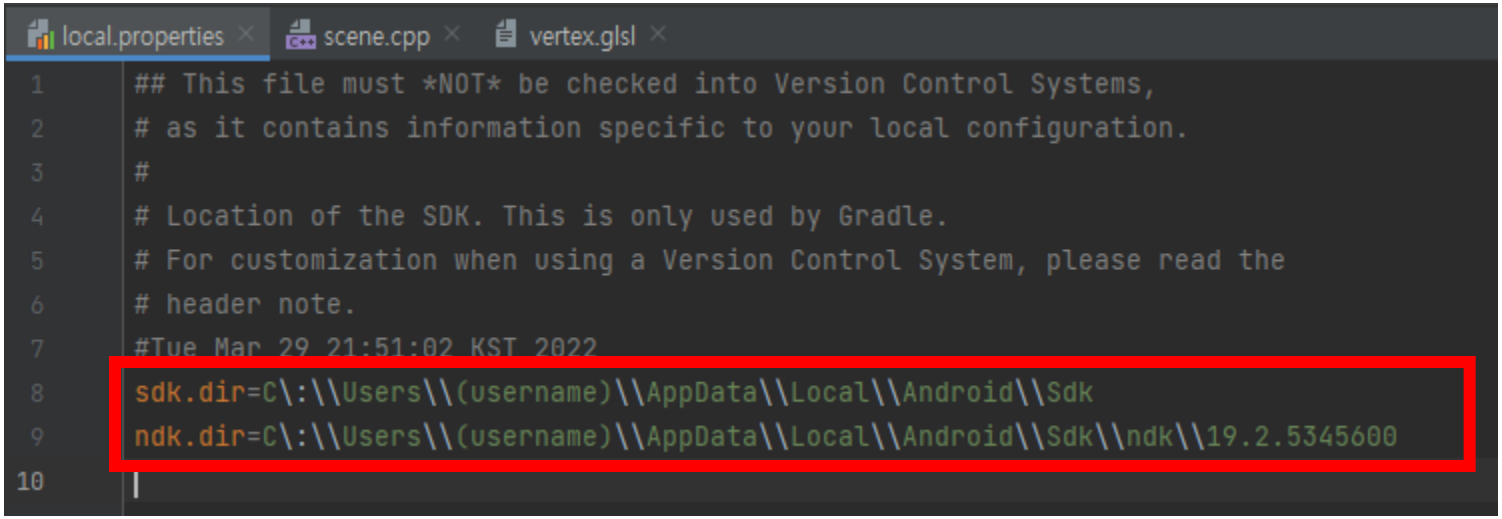
Android SDK

- Use **NDK version 19.2.5345600**.



Gradle Sync

- Before doing homework, you have to modify `local.properties` file.
  - Change (username) of `ndk.dir` and `sdk.dir` to **your PC username**.

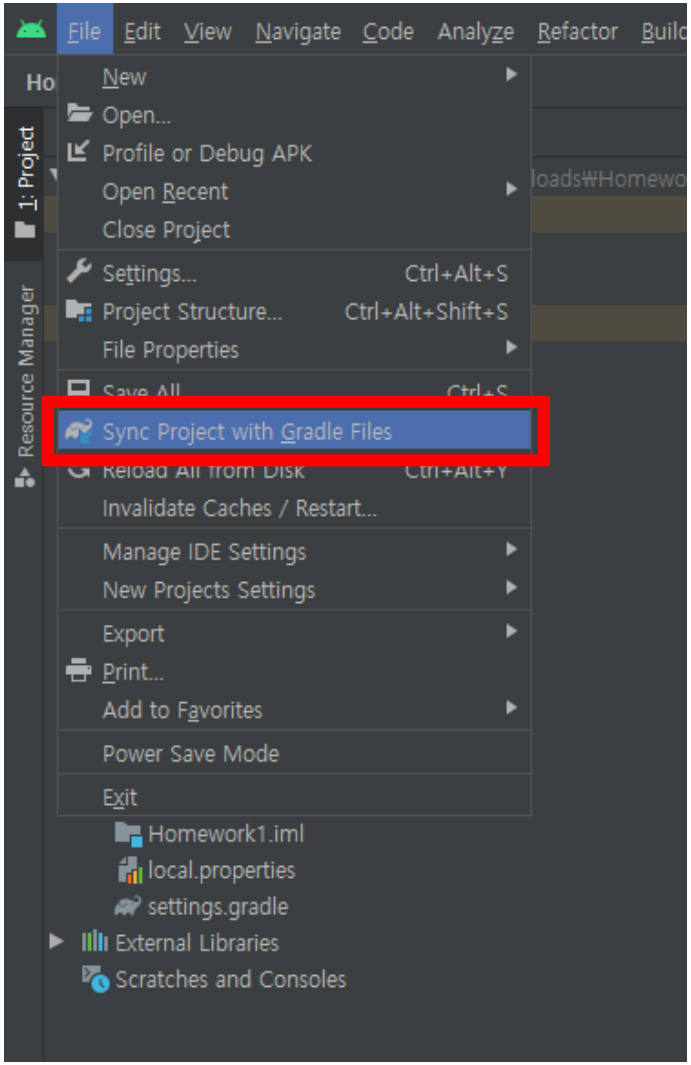


  - Use the following path for MacOS device.

```
sdk.dir=/Users/(username)/Library/Android/sdk
ndk.dir=/Users/(username)/Library/Android/sdk/ndk/19.2.5345600
```

Gradle Sync

- After modifying, sync project with gradle files.
  - Files – Sync Project with Gradle Files



Problem 1

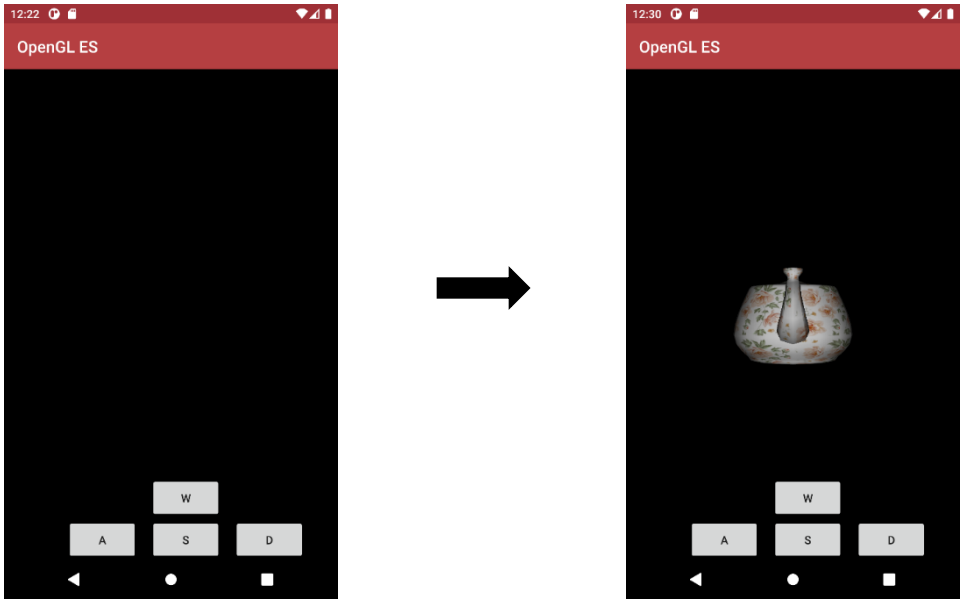
- Fill in some lines in the vertex shader.
  - `app/src/main/assets/vertex.glsl`

vertex.glsl | scene.cpp

```
13
14  void main() {
15
16      ////////////////////////////
17      /* TODO: Problem 1.
18       *  Fill in the lines below.
19       */
20
21      // gl_Position = ;
22      // v_normal = ;
23      // v_texCoord =;
24      ////////////////////////////
25
26      vec3 posWS = (worldMat * vec4(position, 1.0)).xyz;
27      v_lightDir = normalize(lightPos - posWS);
28  }
```

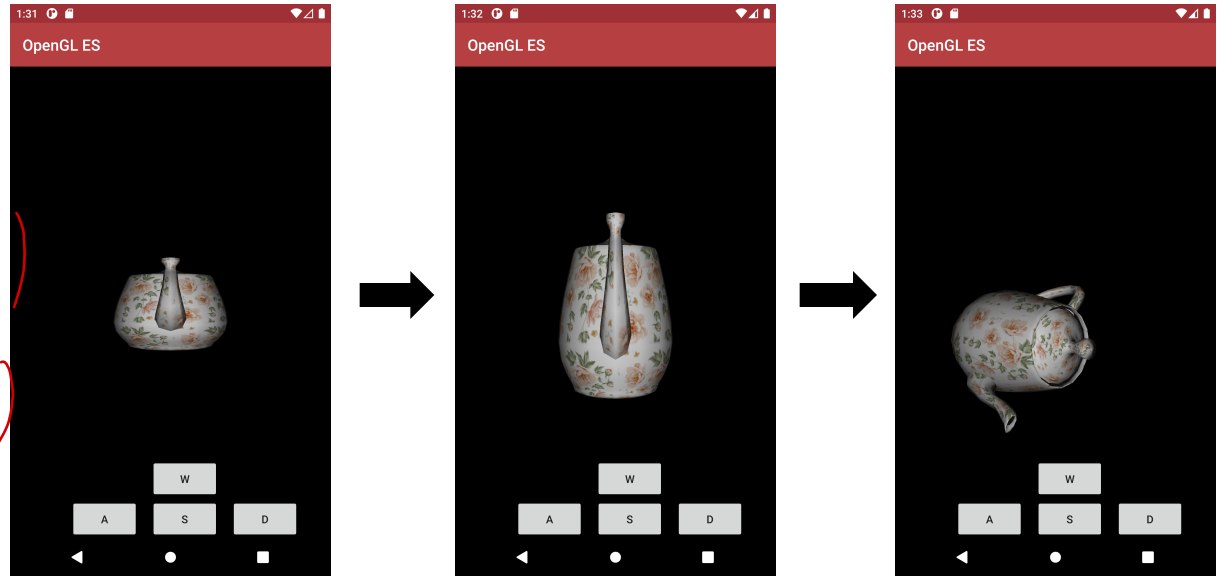
Problem 1

- The clip-space **vertex position** must be calculated.
- The world-space **vertex normal** must be calculated accurately considering non-uniform scaling.
  - Note that the lighting will look weird if vertex normals are incorrect.
- The **texture coordinates** will be sent to the rasterizer stage with no change.



Problem 2

- Transform the teapot as follows:
  1. Scale the teapot by 2.0 along the **y-axis**.
  2. Rotate the teapot by 90° clockwise about the **rotation axis defined by two points (-10, 0, 0) → (0, 0, 10).**
     - Note that an angle can be represented in **degrees** or **radians**.
     - Hint: This rotation can be expressed as the **multiplication of five matrices**.



Problem 2

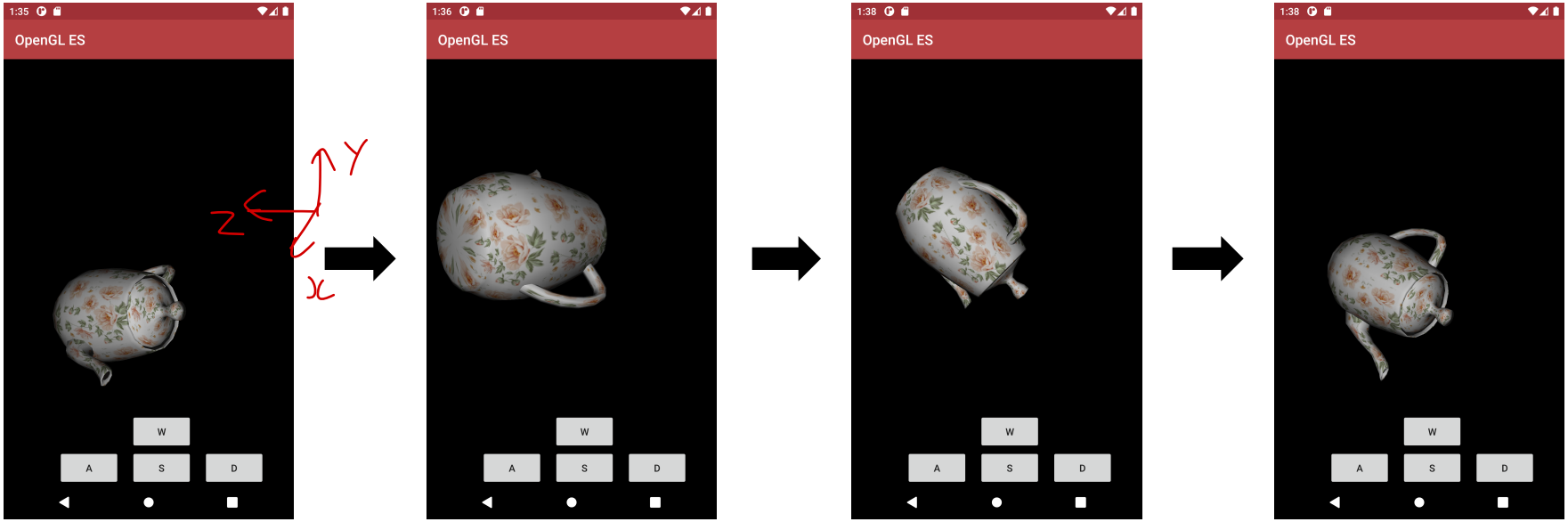
- Fill in some lines in `Scene::setup(…)`.
  - `app/src/main/cpp/scene.cpp`

vertex.glsl | scene.cpp

```
38      // create teapot object
39      teapot = new Object(program, flower, objTeapotVertices, objTeapotIndices,
40                          objTeapotVerticesSize, objTeapotIndicesSize);
41
42      ////////////////////////////
43      /* TODO: Problem 2.
44       *  Scale the teapot by 2.0 along the y-axis.
45       *  Rotate the teapot by 90° CW about the rotation axis defined by two points
46       *  (0, 0, 10) → (10, 0, 20).
47       */
48
49      mat4 scaleM, rotMat;
50      scaleM = transpose( m: mat4( x0: 1.0f,  y0: 0.0f,  z0: 0.0f,  w0: 0.0f,  // In OpenGL, the matrix must be transposed
51                                   x1: 0.0f,  y1: 1.0f,  z1: 0.0f,  w1: 0.0f,
52                                   x2: 0.0f,  y2: 0.0f,  z2: 1.0f,  w2: 0.0f,
53                                   x3: 0.0f,  y3: 0.0f,  z3: 0.0f,  w3: 1.0f));
54
55      // rotMat =;
56      // teapot->worldMatrix =;
57      ////////////////////////////
58  }
59
```

Problem 3

- Rotate the teapot about the **z-axis**.



Problem 3

Problem 3

- Fill a line in `Scene::update(float deltaTime)`.
  - `app/src/main/cpp/scene.cpp`

    `deltaTime` is given in seconds

vertex.glsl | scene.cpp

```
65
66  void Scene::update(float deltaTime) {
67      // use program
68      program->use();
69
70      ////////////////////////////
71      /* TODO: Problem 3.
72       *  Rotate the teapot about the z-axis.
73       */
74
75      // teapot->worldMatrix = ;
76      ////////////////////////////
77
78
79      camera->updateViewMatrix();
80      camera->updateProjectionMatrix();
81      light->setup();
82
83
84      // draw teapot
85      teapot->draw();
86  }
87
```

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Problem 4

- Complete camera movement and rotation.

translate

- Move camera along u, v, n in view matrix when buttons are pressed.
- Rotate camera when the screen is dragged.

Problem 4

Problem 4

- Fill in some lines in `scene.cpp`.
  - `app/src/main/cpp/scene.cpp`

rotation

vertex.glsl | scene.cpp

```
97
98  void Scene::rotateCameraYaw(float theta) {
99
100     ////////////////////////////
101     /* TODO: Problem 4.
102      *  Calculate the rotated u,n vector about v axis.
103      *  Argument theta is amount of rotation in radians. theta is positive when CCW.
104      *  Note that u,v,n should always be orthonormal.
105      *  The u vector can be accessed via camera->cameraU.
106      *  The v vector can be accessed via camera->cameraV.
107      *  The n vector can be accessed via camera->cameraN.
108      */
109
110     camera->updateViewMatrix();
111     ////////////////////////////
112 }
113
114 void Scene::rotateCameraPitch(float theta) {
115
116     ////////////////////////////
117     /* TODO: Problem 4.
118      *  Calculate the rotated v,n vector about u axis.
119      *  Argument theta is amount of rotation in radians. Theta is positive when CCW.
120      *  Note that u,v,n should always be orthonormal.
121      *  The u vector can be accessed via camera->cameraU.
122      *  The v vector can be accessed via camera->cameraV.
123      *  The n vector can be accessed via camera->cameraN.
124      */
125
126     camera->updateViewMatrix();
127     ////////////////////////////
128 }
```

translation

vertex.glsl | scene.cpp

```
130 void Scene::translateLeft(float amount) {
131
132     ////////////////////////////
133     /* TODO: Problem 4.
134      *  Calculate the camera position(eye) when translated left.
135      */
136
137     camera->updateViewMatrix();
138     ////////////////////////////
139 }
140
141 void Scene::translateFront(float amount) {
142
143     ////////////////////////////
144     /* TODO: Problem 4.
145      *  Calculate the camera position(eye) when translated front.
146      */
147
148     camera->updateViewMatrix();
149     ////////////////////////////
150 }
151
152 void Scene::translateRight(float amount) {
153
154     ////////////////////////////
155     /* TODO: Problem 4.
156      *  Calculate the camera position(eye) when translated right.
157      */
158
159     camera->updateViewMatrix();
160     ////////////////////////////
161 }
162
163 void Scene::translateBack(float amount) {
164
165     ////////////////////////////
166     /* TODO: Problem 4.
167      *  Calculate the camera position(eye) when translated back.
168      */
169
170     camera->updateViewMatrix();
171     ////////////////////////////
172 }
```

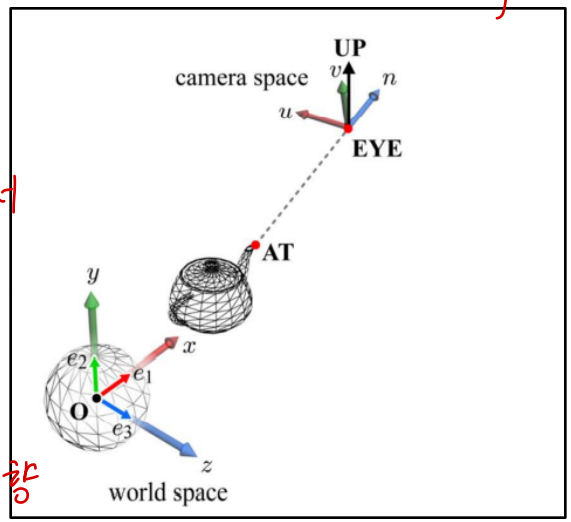
eye 갱신 : camera → eye

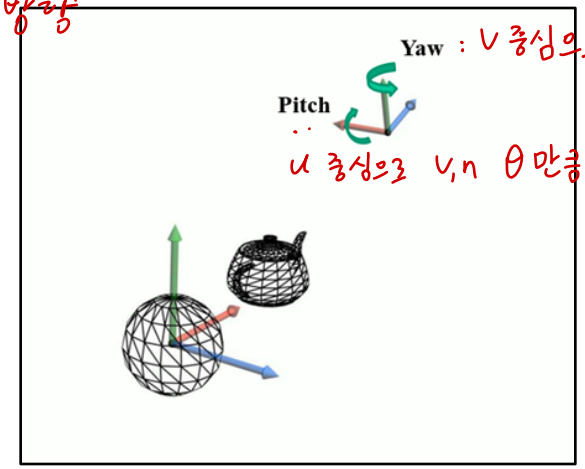
Problem 4

- Functions to fill in:
  - `rotateCameraYaw(float theta)`
    - Horizontal rotation → 변한 u, n을 계산해서 넣어줘야함
  - `rotateCameraPitch(float theta)`
    - Vertical rotation → v, n 〃
  - `translateLeft(float amount)` : u 반대 방향
  - `translateFront(float amount)` : n의 반대 방향
  - `translateRight(float amount)` : u 방향
  - `translateBack(float amount)` : n 방향

CCW 회전 각도

translation 할 양





GLM Library

- You may want to use some glm functions.

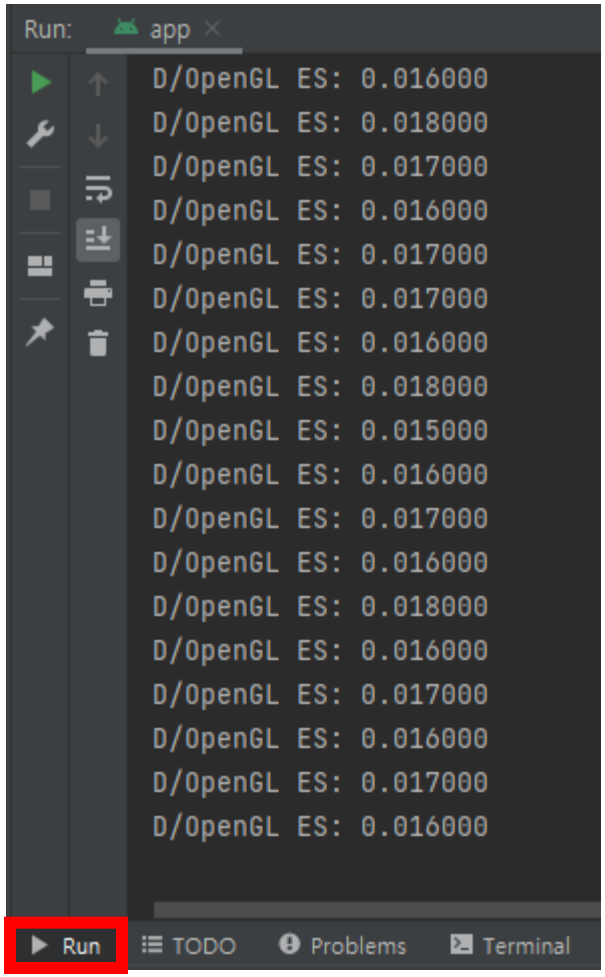
```
glm::cross(…)
glm::normalize(…)
glm::translate(…)
glm::rotate(…)
…
```

Debugging

- You can use `LOG_PRINT_DEBUG()` for debugging purposes

example)

```
69  void Scene::update(float deltaTime) {
70      // use program
71      program->use();
72
73      ////////////////////////////////
74      /* TODO: Problem 3.
75       *  Rotate the teapot about the z-axis.
76       */
77
78      LOG_PRINT_DEBUG("%f", deltaTime);
79
80      // teapot->worldMatrix = ;
81      ////////////////////////////////
82
83
```



Submission

- Deadline
  - April 13 (Wed) 14:00
- Submission files ({student_id}_{name}.zip)
  - Vertex shader file (app/src/main/assets/vertex.glsl)
  - Scene class file (app/src/main/cpp/scene.cpp)
- Submission to Blackboard
- Contact
  - TA email: 2022.CG.TA@gmail.com