

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

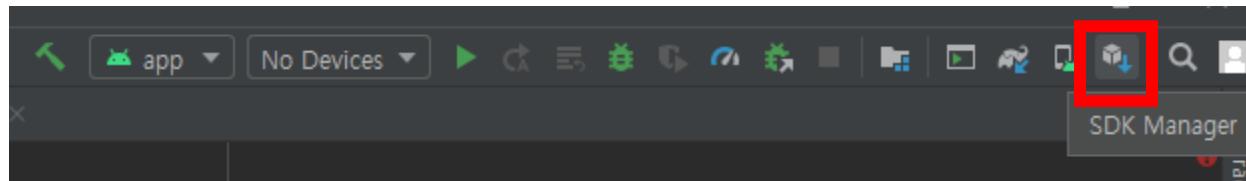
Android Studio

- **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.



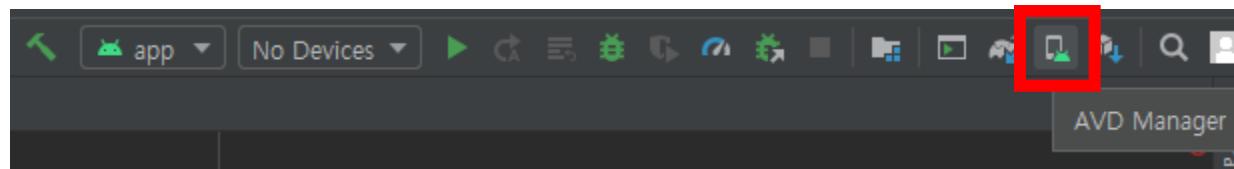
- Use SDK API level 30

The screenshot shows the 'SDK Platforms' tab of the Android Studio SDK Manager. The interface has a dark theme. At the top, there are three tabs: 'SDK Platforms' (which is selected), 'SDK Tools', and 'SDK Update Sites'. Below the tabs, a descriptive text states: 'Each Android SDK Platform package includes the Android platform and sources pertaining to an API level by default. Once installed, the IDE will automatically check for updates. Check "show package details" to display individual SDK components.' A table follows, listing available SDK platforms:

Name	API Level	Revision	Status
<input type="checkbox"/> Android Tiramisu Preview	Tiramisu	2	Not installed
<input type="checkbox"/> Android API 32	32	1	Not installed
<input type="checkbox"/> Android 12.0 (S)	31	1	Not installed
<input checked="" type="checkbox"/> Android 11.0 (R)	30	3	Installed
<input type="checkbox"/> Android 10.0 (Q)	29	5	Not installed
<input type="checkbox"/> Android 9.0 (Pie)	28	6	Not installed
<input type="checkbox"/> Android 8.1 (Oreo)	27	3	Partially installed

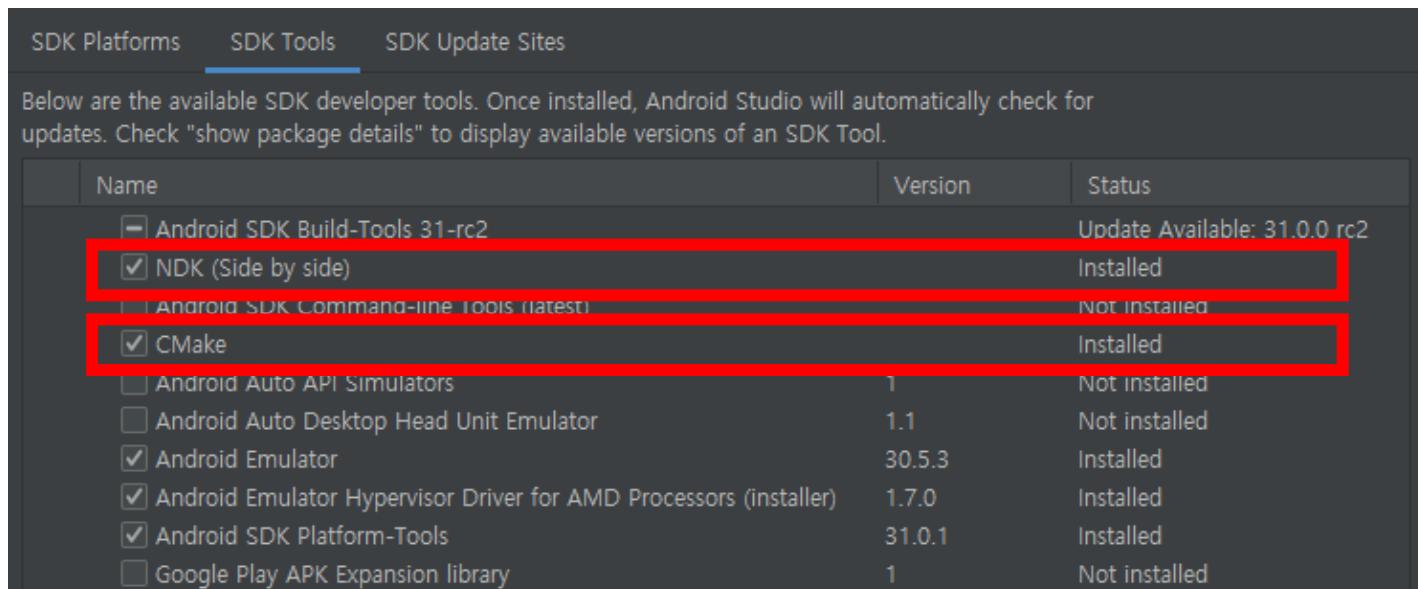
Android SDK

- 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**)



The screenshot shows the 'SDK Tools' tab in the Android Studio SDK Manager. It lists various developer tools with their status: Installed, Not installed, or Update Available. The NDK and CMake entries are highlighted with red boxes.

Name	Version	Status
Android SDK Build-Tools 31-rc2		Update Available: 31.0.0 rc2
NDK (Side by side)		Installed
Android SDK Command-line Tools (latest)		Not installed
CMake		Installed
Android Auto API Simulators	1	Not installed
Android Auto Desktop Head Unit Emulator	1.1	Not installed
Android Emulator	30.5.3	Installed
Android Emulator Hypervisor Driver for AMD Processors (installer)	1.7.0	Installed
Android SDK Platform-Tools	31.0.1	Installed
Google Play APK Expansion library	1	Not installed

Android SDK

- Use NDK version 19.2.5345600.

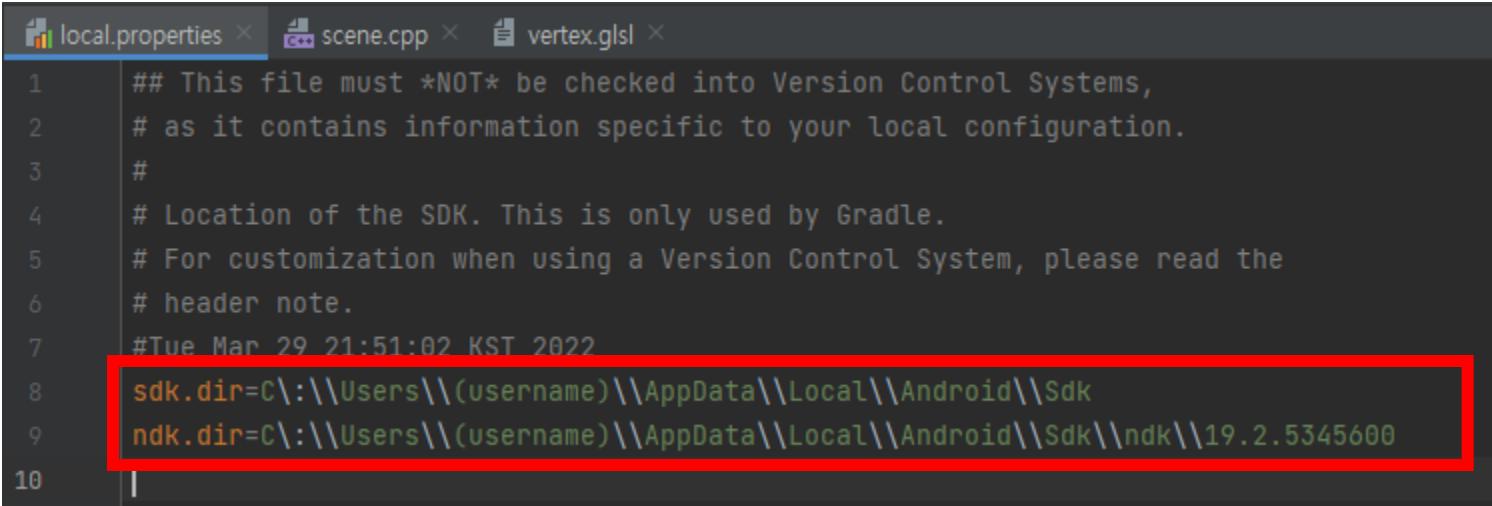
Below are the available SDK developer tools. Once installed, the IDE will automatically check for updates.
Check "show package details" to display available versions of an SDK Tool.

Name	Version	Status
> └ Android SDK Build-Tools 33-rc2		
└ NDK (Side by side)		
<input type="checkbox"/> 24.0.8215888	24.0.8215888	Not installed
<input type="checkbox"/> 23.1.7779620	23.1.7779620	Not installed
<input type="checkbox"/> 23.0.7599858	23.0.7599858	Not installed
<input type="checkbox"/> 22.1.7171670	22.1.7171670	Not installed
<input type="checkbox"/> 22.0.7026061	22.0.7026061	Not installed
<input type="checkbox"/> 21.4.7075529	21.4.7075529	Not installed
<input type="checkbox"/> 21.3.6528147	21.3.6528147	Not installed
<input type="checkbox"/> 21.2.6472646	21.2.6472646	Not installed
<input type="checkbox"/> 21.1.6352462	21.1.6352462	Not installed
<input type="checkbox"/> 21.0.6113669	21.0.6113669	Not installed
<input type="checkbox"/> 20.1.5948944	20.1.5948944	Not installed
<input type="checkbox"/> 20.0.5594570	20.0.5594570	Not installed
<input checked="" type="checkbox"/> 19.2.5345600	19.2.5345600	Installed
<input type="checkbox"/> 18.1.5063045	18.1.5063045	Not installed
<input type="checkbox"/> 17.2.4988734	17.2.4988734	Not installed
<input type="checkbox"/> 16.1.4479499	16.1.4479499	Not installed
└ Android SDK Command-line Tools (latest)		

Hide Obsolete Packages Show Package Details

Gradle Sync

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



```
local.properties x scene.cpp x vertex.glsl x
1 ## This file must *NOT* be checked into Version Control Systems,
2 # as it contains information specific to your local configuration.
3 #
4 # Location of the SDK. This is only used by Gradle.
5 # For customization when using a Version Control System, please read the
6 # header note.
7 #Tue Mar 29 21:51:02 KST 2022
8 sdk.dir=C:\\\\Users\\\\(username)\\\\AppData\\\\Local\\\\Android\\\\Sdk
9 ndk.dir=C:\\\\Users\\\\(username)\\\\AppData\\\\Local\\\\Android\\\\Sdk\\\\ndk\\\\19.2.5345600
10 |
```

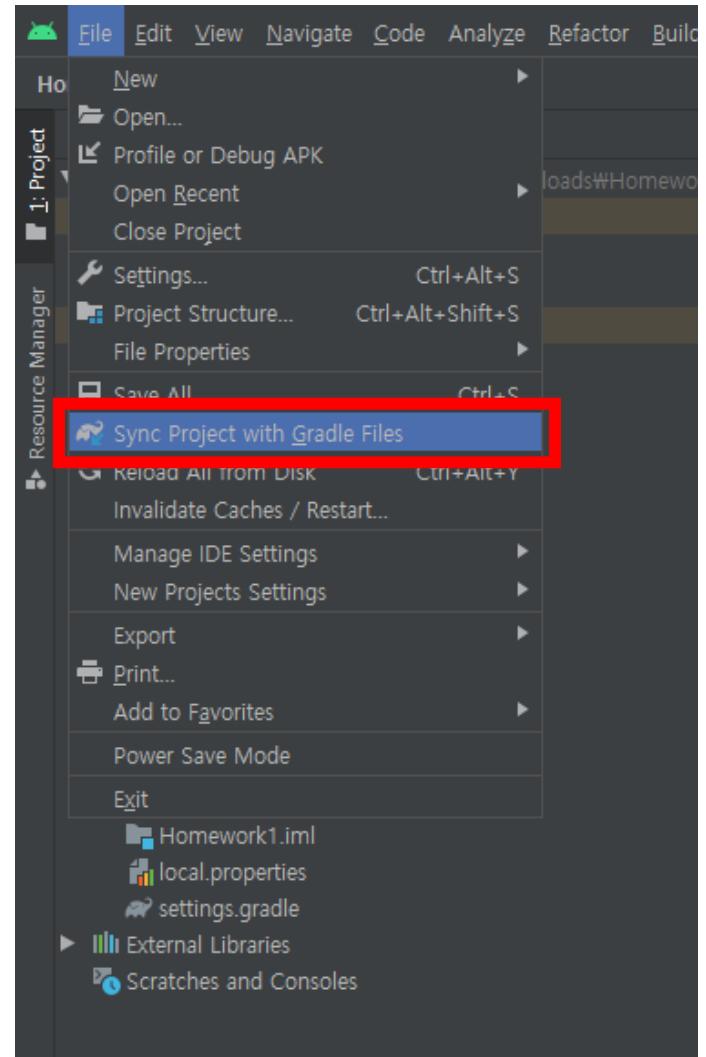
- Use the following path for MacOS device.

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

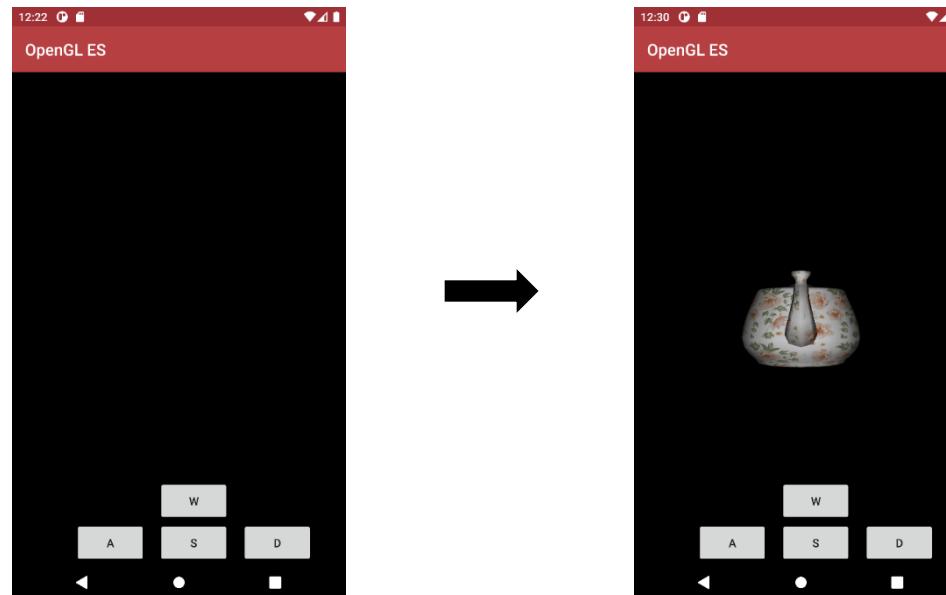
- Fill in some lines in the vertex shader.

- 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         // vTexCoord =;
24         /////////////
25
26         vec3 posWS = (worldMat * vec4(position, 1.0)).xyz;
27         v_lightDir = normalize(lightPos - posWS);
28 }
```

Problem 1

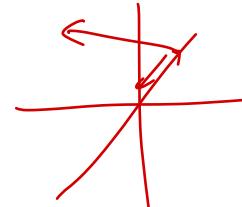
- The clip-space **vertex position** must be calculated.
- The world-space **vertex normal** must be calculated accurately considering non-uniform scaling.
 - Note that the lighting will look weird if vertex normals are incorrect.
- The **texture coordinates** will be sent to the rasterizer stage with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



Problem 2

$$\begin{pmatrix} 1 & 0 & 0 & -10 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & -1 & 0 & 0 \\ \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & -1 & 0 & 0 \\ \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 10 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} \frac{1}{2} & \frac{\sqrt{2}}{2} & \frac{1}{2} & -5 \\ \frac{\sqrt{2}}{2} & 0 & \frac{\sqrt{2}}{2} & -5\sqrt{2} \\ \frac{1}{2} & -\frac{\sqrt{2}}{2} & \frac{1}{2} & 5 \end{pmatrix}$$



- Transform the teapot as follows:

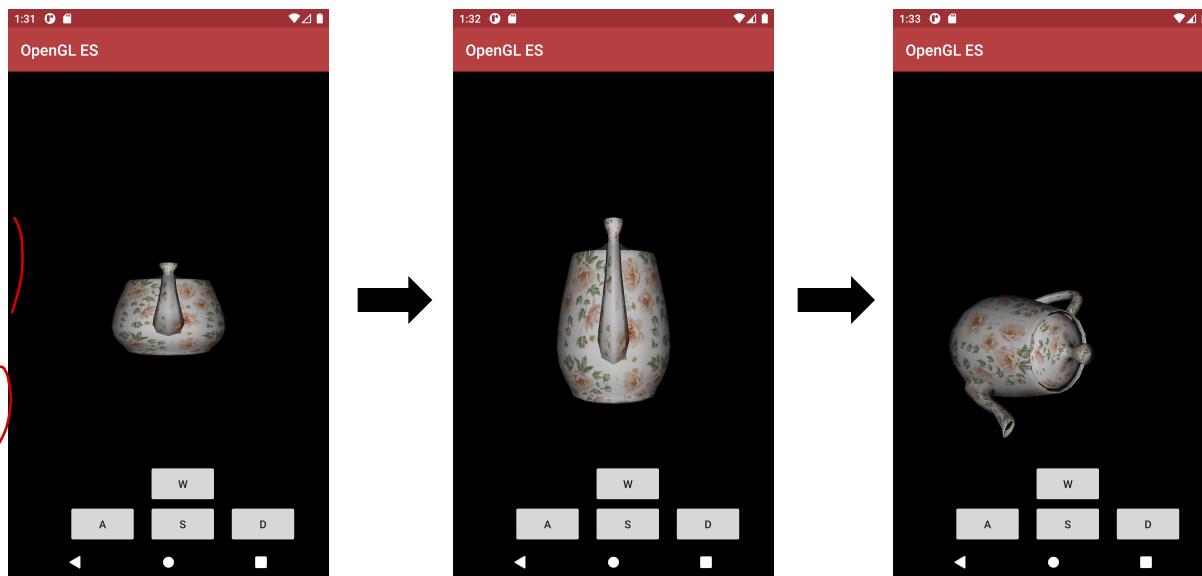
- Scale the teapot by 2.0 along the y-axis.
- Rotate the teapot by 90° clockwise about the rotation axis defined by two points

$$(-10, 0, 0) \rightarrow (0, 0, 10).$$

$$T^{-1} R_i R_{y_0} R_i T \quad \text{axis } (10, 0, 10)$$

- Note that an angle can be represented in **degrees or radians**.
- Hint: This rotation can be expressed as the **multiplication of five matrices**.

$$T(+10, 0, 0)$$



$$u = \left(-\frac{1}{\sqrt{2}}, 0, \frac{1}{\sqrt{2}} \right)$$

$$v = (0, -1, 0)$$

$$n = \left(\frac{1}{\sqrt{2}}, 0, \frac{1}{\sqrt{2}} \right)$$

Problem 2

- Fill in some lines in `Scene::setup(...)`.

- `app/src/main/cpp/scene.cpp`

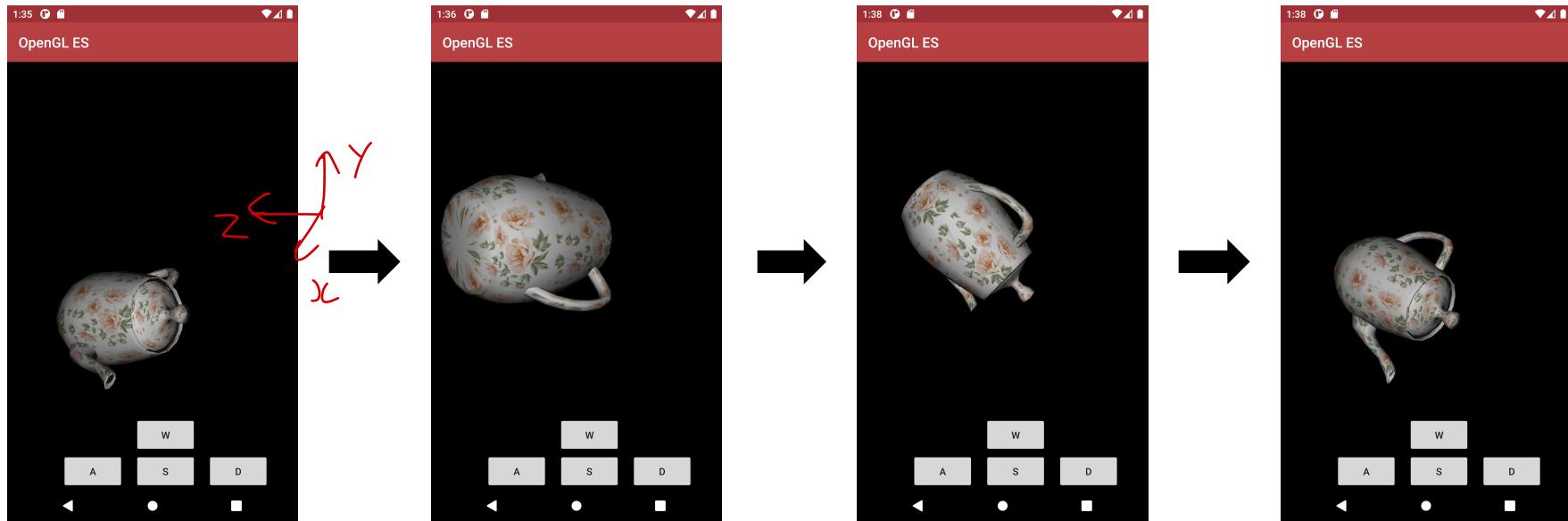
```
// create teapot object
teapot = new Object(program, flower, objTeapotVertices, objTeapotIndices,
                    objTeapotVerticesSize, objTeapotIndicesSize);

/////////////////////////////
/* TODO: Problem 2.
 * Scale the teapot by 2.0 along the y-axis.
 * Rotate the teapot by 90° CW about the rotation axis defined by two points
 * (0, 0, 10) → (10, 0, 20).
 */

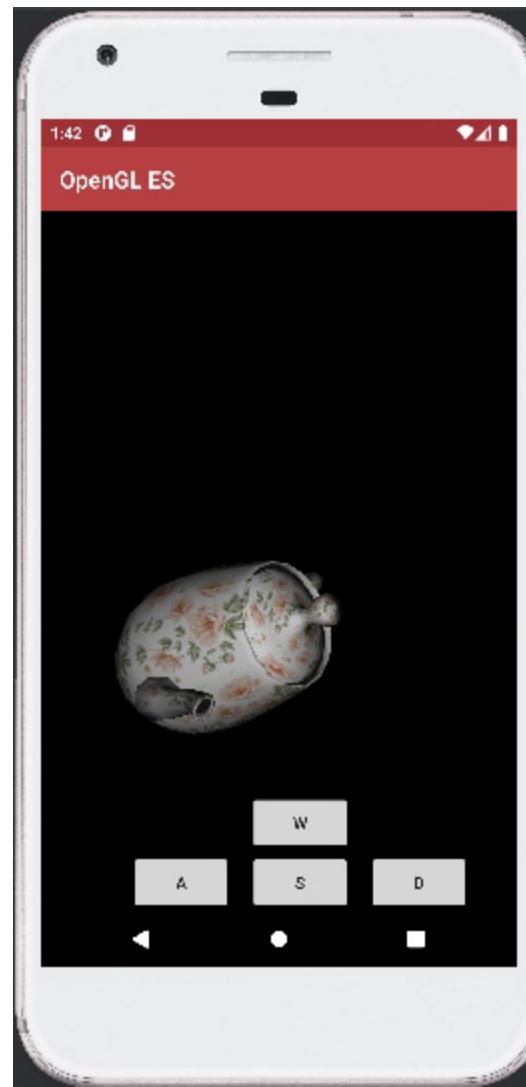
mat4 scaleM, rotMat;
scaleM = transpose( mat4( x0: 1.0f, y0: 0.0f, z0: 0.0f, w0: 0.0f, // In OpenGL, the matrix must be transposed
                           x1: 0.0f, y1: 1.0f, z1: 0.0f, w1: 0.0f,
                           x2: 0.0f, y2: 0.0f, z2: 1.0f, w2: 0.0f,
                           x3: 0.0f, y3: 0.0f, z3: 0.0f, w3: 1.0f));
// rotMat =;  $T^{-1}R^T R_z RT$ 
// teapot->worldMatrix =;
/////////////////////////
```

Problem 3

- Rotate the teapot about the z-axis.



Problem 3

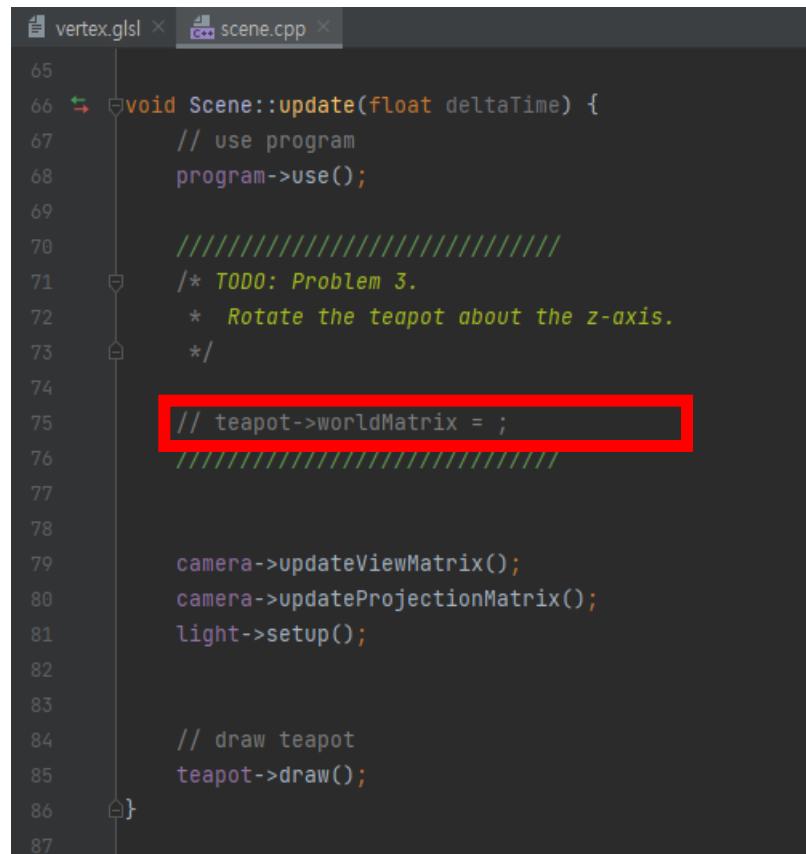


Problem 3

- Fill a line in `Scene::update(float deltaTime)`.

- `app/src/main/cpp/scene.cpp`

`deltaTime` is given in seconds



```
vertex.glsl x scene.cpp x
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
```

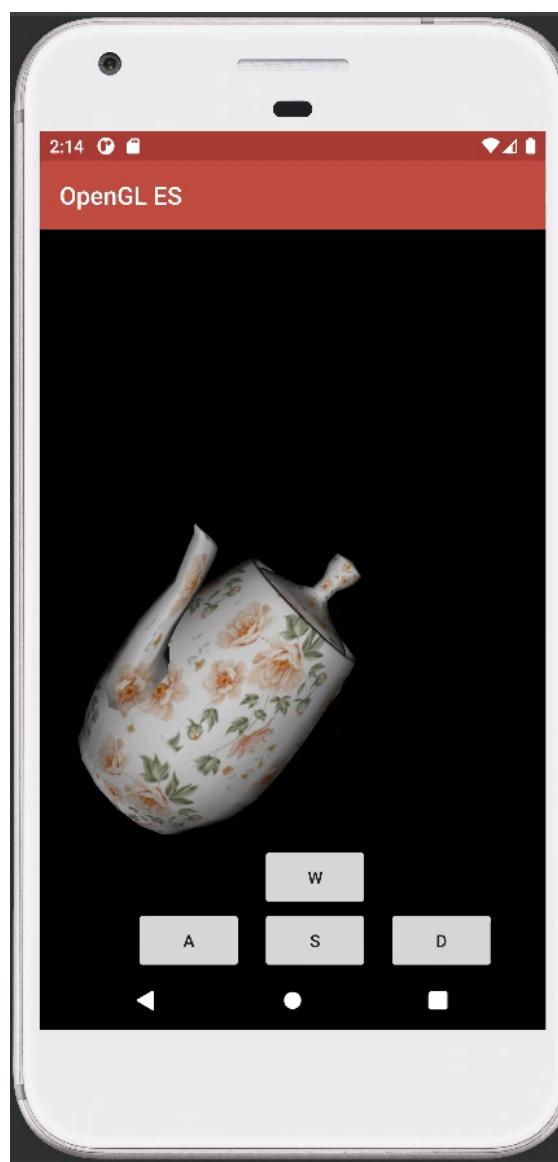
$$\begin{matrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{matrix}$$

Problem 4

+translate

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

Problem 4

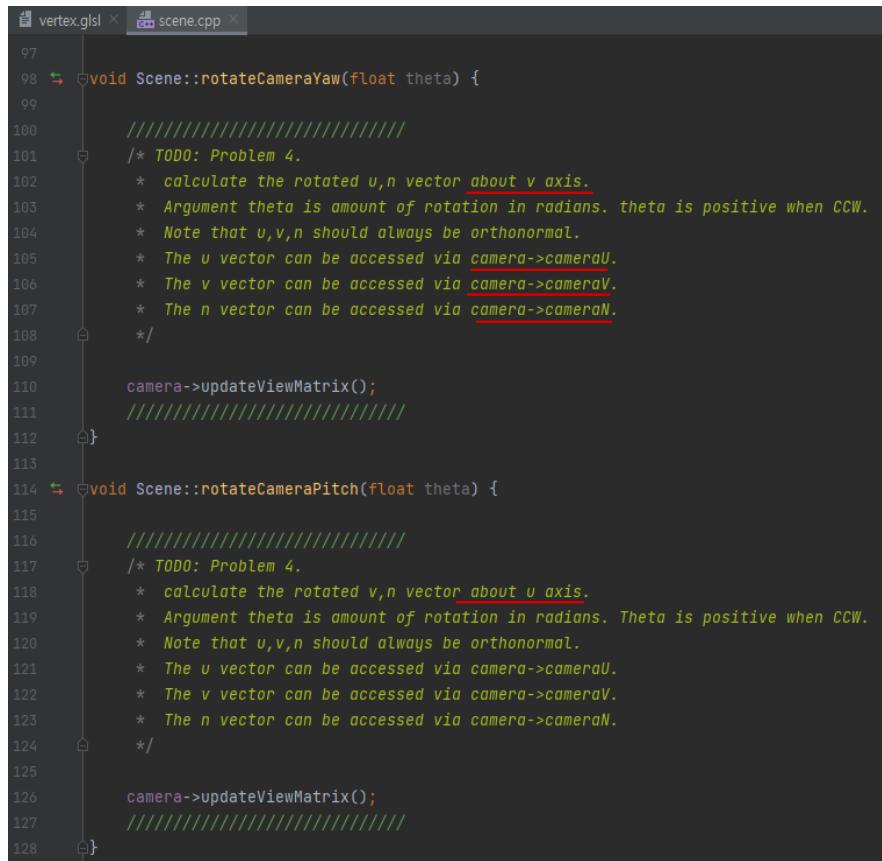


Problem 4

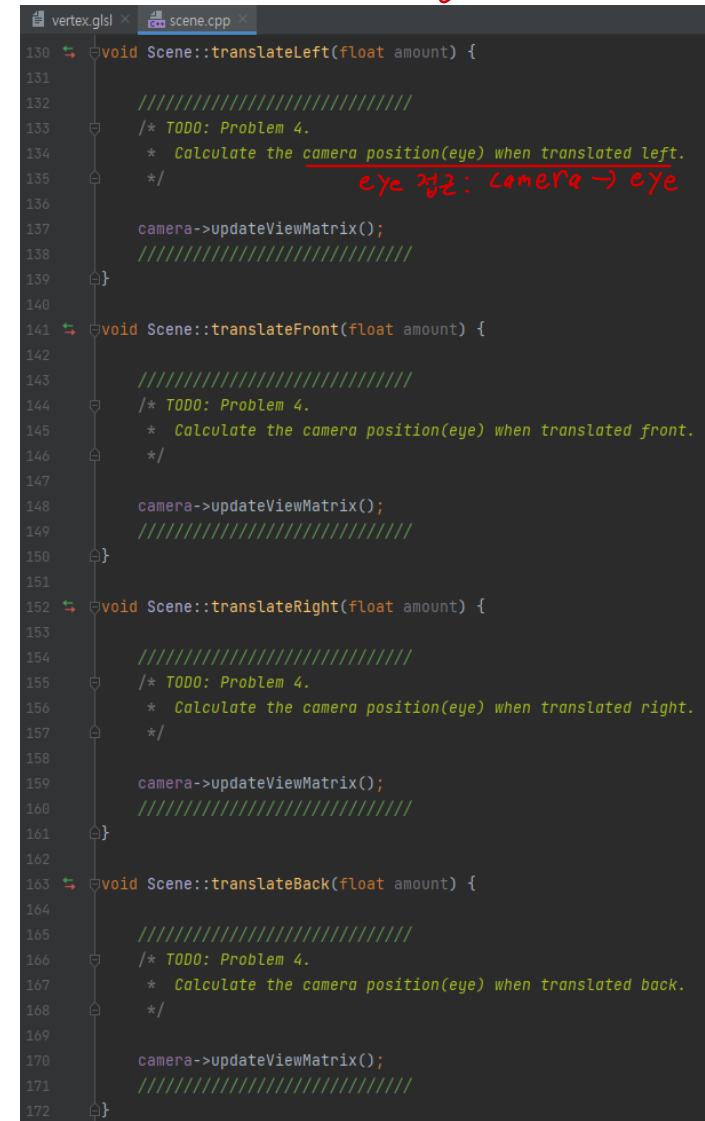
- Fill in some lines in `scene.cpp`.

- app/src/main/cpp/`scene.cpp`

↙ rotation



```
vertex.glsl x scene.cpp x
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 x scene.cpp x
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 }
```

Problem 4

$$\begin{bmatrix} 0 & \circ & \circ \\ 0 & - & - \\ \leftarrow & \sim & \sim \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

- Functions to fill in:

- rotateCameraYaw(float theta)

• Horizontal rotation → 번한 u, n 을 계산해서 넣어줘야함

- rotateCameraPitch(float theta)

• Vertical rotation → v, n "

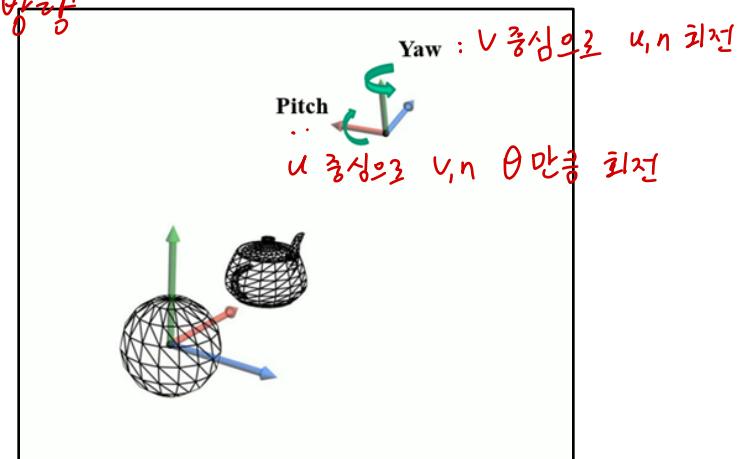
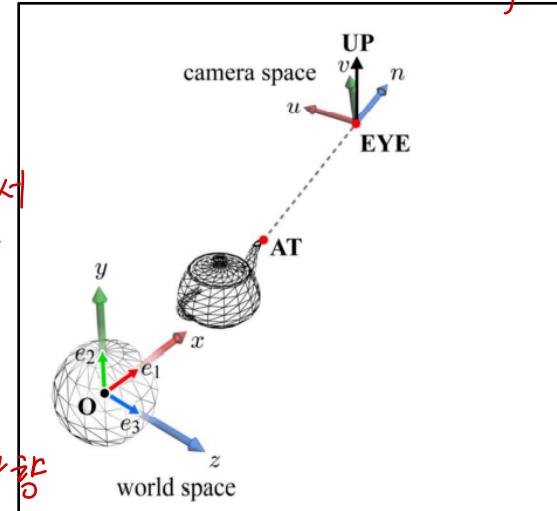
- translateLeft(float amount) : u 반대 방향

- translateFront(float amount) : v 의 반대 방향

- translateRight(float amount) : u 방향

- translateBack(float amount) : v 방향

↓
translation
값 양



GLM Library

- You may want to use some glm functions. *아이거 쓰면 고드 짱짱*

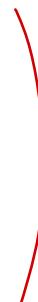
`glm::cross(...)`

`glm::normalize(...)`

`glm::translate(...)`

`glm::rotate(...)`

...



매개변수 정색

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.  
     * Rotate the teapot about the z-axis.  
     */  
75  
76      LOG_PRINT_DEBUG("%f", deltaTime);  
77  
78      // teapot->worldMatrix = ;  
79      ////////////////////////  
80  
81  
82  
83
```

The screenshot shows an IDE interface with a terminal window titled "Run: app". The terminal displays a series of identical log entries: "D/OpenGL ES: 0.016000" repeated 20 times. At the bottom of the terminal window, there is a navigation bar with several icons: a play button (Run), a list icon (TODO), a gear icon (Problems), and a terminal icon (Terminal). A red rectangular box highlights the "Run" button icon.

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