Game Graphic Programming

Homework1

IIIXR Lab. SeungJeh Chung

teclados078@khu.ac.kr

Goal

Goal

- Render three objects (cube, person, and teapot).
- Animate the rendered objects.

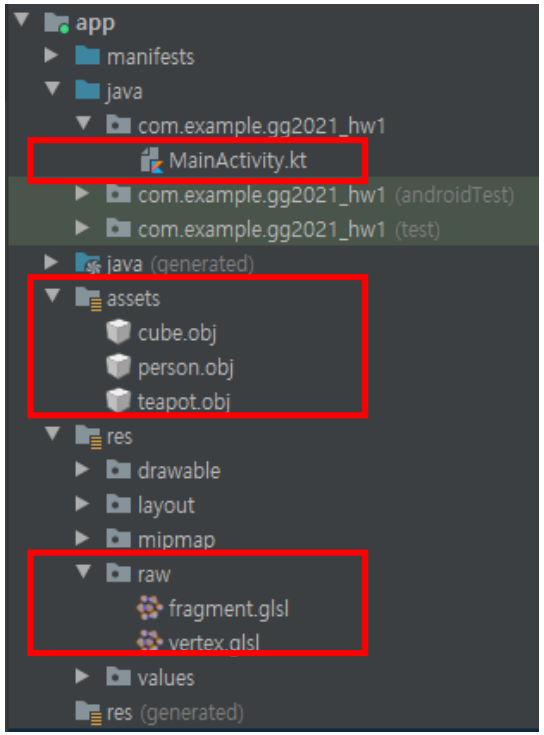
Detail

- Render three objects in a single scene.
- Compute world matrix.
- Compute view matrix and projection matrix via appropriate functions.
- Write and load the vertex.glsl and fragment.glsl.
- Animate the objects every frame. (rotation and scaling)

Problem

1. Project setting

- Project name: GG2021_HW1_학번 (ex. GG2021_HW1_2017103754)
- Project directory:



Problem

2. Object class

- Modify an object class to manage all three objects.

```kotlin
class MyGLRenderer(context: Context): GLSurfaceView.Renderer{
    private val mContext: Context = context
    private var vPMatrix = FloatArray( size: 16)
    private var projectionMatrix = FloatArray( size: 16)
    private var viewMatrix = FloatArray( size: 16)
    //P. model matrix & 매 프레임 변화 matrix 선언

    //P. object 선언
    private lateinit var cube: Cube


    override fun onSurfaceCreated(p0: GL10?, p1: EGLConfig?) {
        GLES20.glClearColor(0.0f, 0.0f, 0.0f, 1.0f)
        //P. object 초기화
        cube = Cube(mContext)

        //P. model matrix & 매 프레임 변화 matrix 초기화

    }

    override fun onDrawFrame(p0: GL10?) {
        GLES20.glClear(GLES20.GL_COLOR_BUFFER_BIT)

        //P. 아래 구현한 mySetLookAtM function 으로 수정
        Matrix.setLookAtM(viewMatrix, rmOffset: 0, eyeX: 1.0f, eyeY: 1.0f, eyeZ: -3f, centerX: 0f, ce

        //P. 각 object 별 매 프레임 변화 matrix 와 model matrix 를 multiply

        Matrix.multiplyMM(vPMatrix, 0, projectionMatrix, 0, viewMatrix, 0)

        //P. object draw
        cube.draw(vPMatrix)
    }
```

```kotlin
//PP. cube, person, teapot 모두 포함할 수 있는 Object class 로 수정
class Cube(context: Context){

    //P. 아래 shader code string 지우고, res/raw 에 위치한 vertex.glsl , fragment.glsl 로드해서 vertex
    private val vertexShaderCode =
            "uniform mat4 uMVPMatrix;" +
                    "attribute vec4 vPosition;" +
                    "void main() {" +
                    "  gl_Position = uMVPMatrix * vPosition;" +
                    "}"

    private val fragmentShaderCode =
            "precision mediump float;" +
                    "uniform vec4 vColor;" +
                    "void main() {" +
                    "  gl_FragColor = vColor;" +
                    "}"

    //P. model matrix handle 변수 추가 선언
    private var vPMatrixHandle: Int = 0

    val color = floatArrayOf(1.0f, 0.980392f, 0.980392f, 0.3f)

    private var mProgram: Int

    private var vertices = mutableListOf<Float>()
    private var faces = mutableListOf<Short>()
    private lateinit var verticesBuffer: FloatBuffer
    private lateinit var facesBuffer: ShortBuffer
```

⋮

The part that starts with "//PP" means that you need to find and modify some of the codes in the class under that comment.

Problem

2. Object class

- Modify an object class to manage all three objects.

```
//PP. cube, person, teapot 의 world transform 및 매 프레임 변화를 반영할 수 있는 draw function 으로 수정
fun draw(mvpMatrix: FloatArray){
    GLES20.glUseProgram(mProgram)

    positionHandle = GLES20.glGetAttribLocation(mProgram, "vPosition").also { it: Int
        GLES20.glEnableVertexAttribArray(it)
        GLES20.glVertexAttribPointer(
                it,
                COORDS_PER_VERTEX,
                GLES20.GL_FLOAT,
                normalized: false,
                vertexStride,
                verticesBuffer
        )

        mColorHandle = GLES20.glGetUniformLocation(mProgram, "vColor").also { colorHandle ->
            GLES20.glUniform4fv(colorHandle, 1, color, 0)
        }

        vPMatrixHandle = GLES20.glGetUniformLocation(mProgram, "uMVPMatrix")
        GLES20.glUniformMatrix4fv(vPMatrixHandle, 1, false, mvpMatrix, 0)

        GLES20.glDrawElements(GLES20.GL_TRIANGLES, faces.size, GLES20.GL_UNSIGNED_SHORT, facesBuffer)

        GLES20.glDisableVertexAttribArray(it)
    }
}
```

The part that starts with "//PP" means that you need to find and modify some of the codes in the function under that comment.

Problem

3. World matrix

- Compute the world matrix for each object based on the following conditions:

	Scale	Translate	rotate
Cube	x(0.5), z(0.5)	X	X
Person	X	x(2)	X
Teapot	x, y, z(0.2)	x(1.25), y(0.4)	Y-axis(180 degree)

```
class MyGLRenderer(context: Context): GLSurfaceView.Renderer{
    private val mContext: Context = context
    private var vPMatrix = FloatArray( size: 16)
    private var projectionMatrix = FloatArray( size: 16)
    private var viewMatrix = FloatArray( size: 16)
    //P. model matrix & 매 프레임 변화 matrix 선언

    //P. object 선언
    private lateinit var cube: Cube


    override fun onSurfaceCreated(p0: GL10?, p1: EGLConfig?) {
        GLES20.glClearColor(0.0f, 0.0f, 0.0f, 1.0f)
        //P. object 초기화
        cube = Cube(mContext)

        //P. model matrix & 매 프레임 변화 matrix 초기화

    }
```

```
    override fun onDrawFrame(p0: GL10?) {
        GLES20.glClear(GLES20.GL_COLOR_BUFFER_BIT)

        //P. 아래 구현한 mySetLookAtM function 으로 수정
        Matrix.setLookAtM(viewMatrix,  rmOffset: 0,  eyeX: 1.0f,  eyeY: 1.0f,  eyeZ: -3f,  centerX: 0f,  ce

        //P. 각 object 별 매 프레임 변화 matrix 와 model matrix 를 multiply

        Matrix.multiplyMM(vPMatrix, 0, projectionMatrix, 0, viewMatrix, 0)

        //P. object draw
        cube.draw(vPMatrix)
    }
```

Problem

4. Camera & Projection matrix

- Implement function that computes view matrix & projection matrix.
- Do not use Matrix library function like setLookAtM() and frustumM() (multiplyMM () can be used).
- Condition:
  - View: eye(1.5f, 1.5f, -9f), at(0f, 0f, 0f), up(0f, 1f, 0f) /
  - projection: aspect(screen width/height), fov(60 degree), near(2f), far(12f)

```kotlin
override fun onDrawFrame(p0: GL10?) {
    GLES20.glClear(GLES20.GL_COLOR_BUFFER_BIT)

    //P. 아래 구현한 mySetLookAtM function 으로 수정
    Matrix.setLookAtM(viewMatrix, rmOffset: 0, eyeX: 1.0f, eyeY: 1.0f, eyeZ: -3f, centerX: 0f,

    //P. 각 object 별 매 프레임 변화 matrix 와 model matrix 를 multiply

    Matrix.multiplyMM(vPMatrix, 0, projectionMatrix, 0, viewMatrix, 0)

    //P. object draw
    cube.draw(vPMatrix)
}

override fun onSurfaceChanged(p0: GL10?, width: Int, height: Int) {
    GLES20.glViewport(0,0, width, height)

    val ratio: Float = width.toFloat() / height.toFloat()

    //P.  아래 구현한 myFrustumM function 으로 수정
    Matrix.frustumM(projectionMatrix, offset: 0, -ratio, ratio, bottom: -1f, top: 1f, near: 2f,
}
```

```kotlin
//P. vecNormalize function 구현: 벡터 정규화 함수 (mySetLookAtM function 구현 시 사용)

//P. mySetLookAtM function 구현: viewMatrix 구하는 함수 (Matrix library function 중 multiplyMM 만 사용 가능)

//P. myFrustumM function 구현: projectionMatrix 구하는 함수 (Matrix library function 중 multiplyMM 만 사용 가능)
```

Problem

5. GLSL

- Write and load the GLSL code(vertex.glsl and fragment.glsl).
- Please follow the comments in the mainactivity file.

```
inActivity.kt   vertex.glsl   fragment.glsl
//P. vertex.glsl code 작성
```

```
nActivity.kt   vertex.glsl   fragment.glsl
//P. fragment.glsl code 작성
```

```
//PP. cube, person, teapot 모두 포함할 수 있는 Object class 로 수정
class Cube(context: Context){

    //P. 아래 shader code string 지우고, res/raw 에 위치한 vertex.glsl , fragment.glsl 로드해서 vertexShaderCode, fragmentShaderCode 에 넣기
    private val vertexShaderCode =
            "uniform mat4 uMVPMatrix;" +
                    "attribute vec4 vPosition;" +
                    "void main() {" +
                    "  gl_Position = uMVPMatrix * vPosition;" +
                    "}"

    private val fragmentShaderCode =
            "precision mediump float;" +
                    "uniform vec4 vColor;" +
                    "void main() {" +
                    "  gl_FragColor = vColor;" +
                    "}"
```

Problem

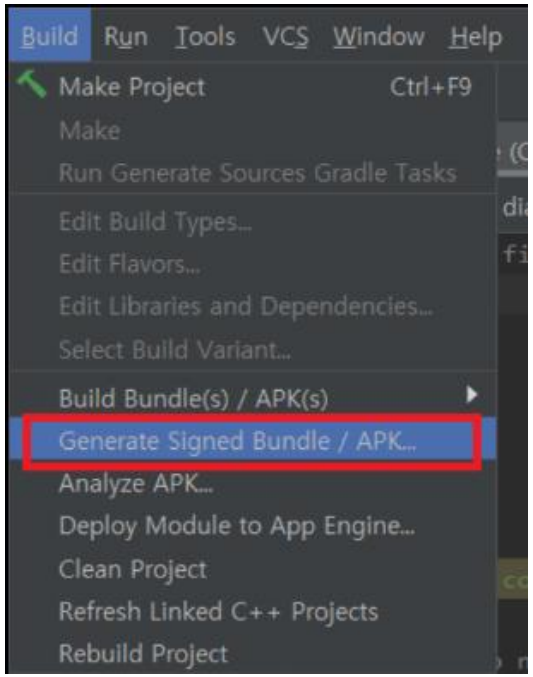
6. Animate the objects every frame.

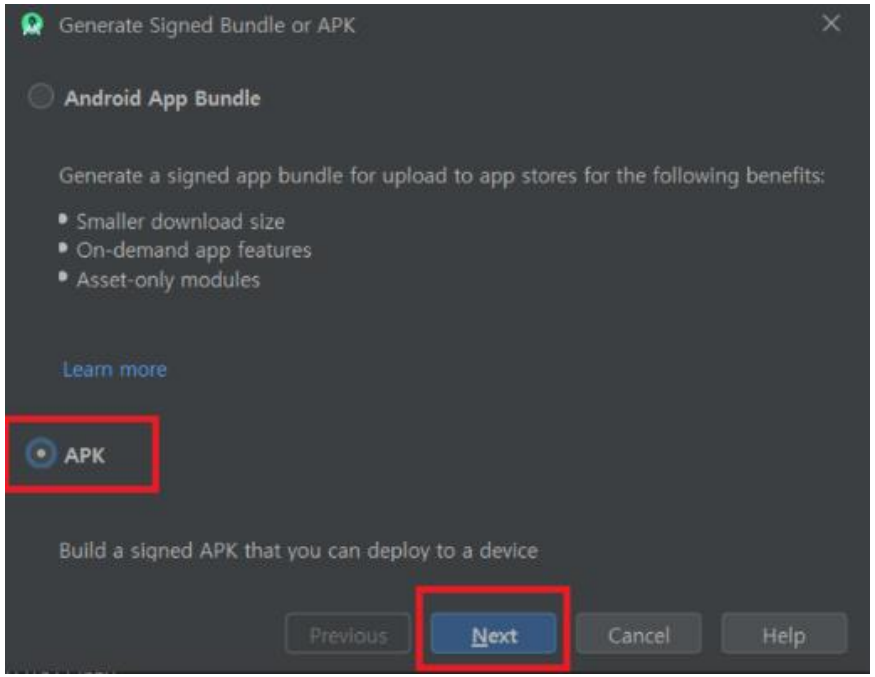
- Animation detail of each object is as follow:
  - *The size of the cube increases by 0.001 times for x- and z-axis, while it increases 0.002 times for the y-axis for every frame.*
  - *If height of the cube exceeds 3.0, it will not grow larger.*
  - *Person and teapot rotate 0.8 degrees per frame around the y-axis.*

```kotlin
class MyGLRenderer(context: Context): GLSurfaceView.Renderer{
    private val mContext: Context = context
    private var vPMatrix = FloatArray( size: 16)
    private var projectionMatrix = FloatArray( size: 16)
    private var viewMatrix = FloatArray( size: 16)
    //P. model matrix & 매 프레임 변화 matrix 선언

    //P. object 선언
    private lateinit var cube: Cube


    override fun onSurfaceCreated(p0: GL10?, p1: EGLConfig?) {
        GLES20.glClearColor(0.0f, 0.0f, 0.0f, 1.0f)
        //P. object 초기화
        cube = Cube(mContext)

        //P. model matrix & 매 프레임 변화 matrix 초기화

    }
```

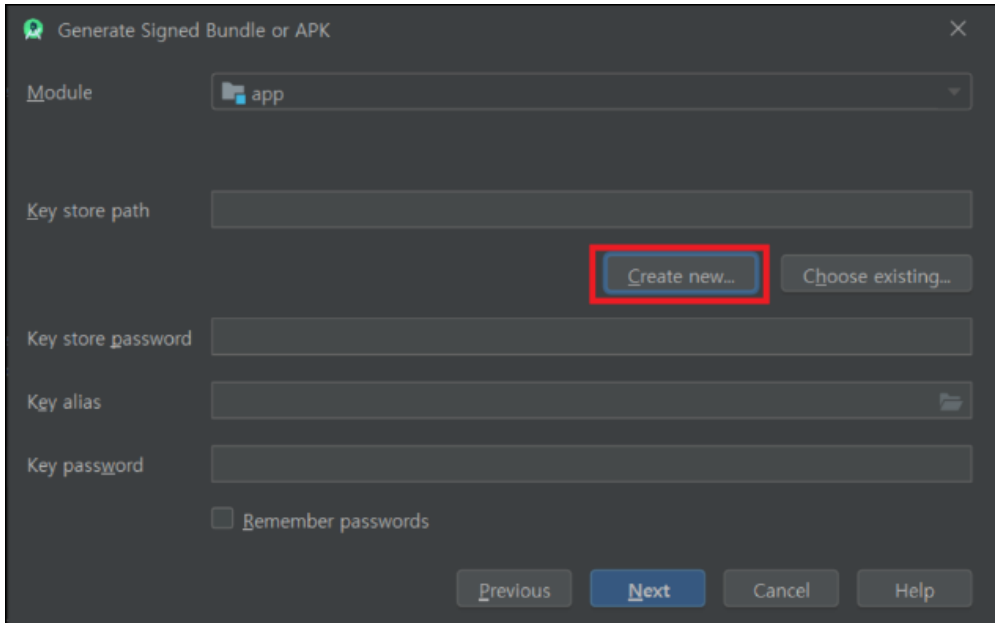
```kotlin
    override fun onDrawFrame(p0: GL10?) {
        GLES20.glClear(GLES20.GL_COLOR_BUFFER_BIT)

        //P. 아래 구현한 mySetLookAtM function 으로 수정
        Matrix.setLookAtM(viewMatrix,  rmOffset: 0,  eyeX: 1.0f,  eyeY: 1.0f,  eyeZ: -3f,  centerX: 0f,  ce

        //P. 각 object 별 매 프레임 변화 matrix 와 model matrix 를 multiply

        Matrix.multiplyMM(vPMatrix, 0, projectionMatrix, 0, viewMatrix, 0)

        //P. object draw
        cube.draw(vPMatrix)
    }
```

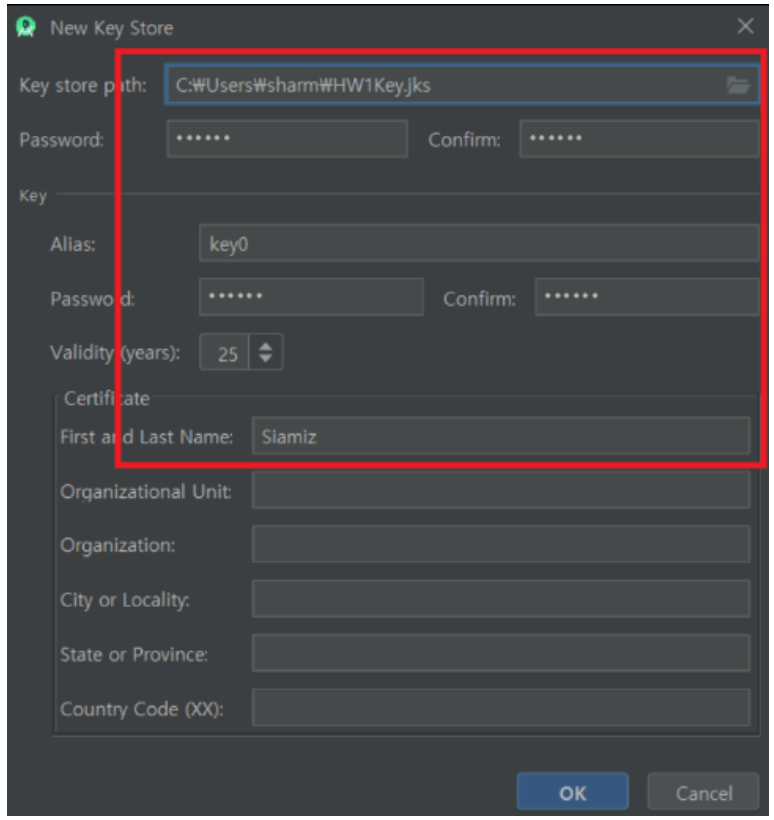
Generate APK





Generate APK





Submission

Deadline

- 05.05. 23:59

Submit followings to e-campus

- Make an apk file and upload it to your git repository. Then, share your git URL and password via e-campus.
- Submit to e-campus (8week.Homework1): git URL text file, GG2021_HW1_학번.zip (MainActivity.kt, vertex.glsl, fragment.glsl)

TA

- 정승재 (teclados078@khu.ac.kr)