Homework 2

COSE331 Computer Graphics

Goal

- Implementing forward kinematics
- Apply skinning on the mesh
- Interpolate the animation between key frames
- Implement Object rotating (optional)

Initial state

- Character is in T-pose.

Final Result

- Character run through the screen.

Final Result (optional)

- Character rotates if you implements object rotating. (optional)

Skeleton data

- The skeleton data will be provided in inc/binary/skeleton.h header file.
  - It has 28 joints including the root.
  - jNames[i] : the name of i-th joint
  - jParents[i] : the index of the parent of i-th joint
  - jOffset[i] : the offset between i-th joint and its parent joint

↳ translation만 저장?

Animation data

- The animation data will be provided in inc/binary/animation.h header file.
  - The animation has 4 key frames.
  - 0 → 1 → 2 → 3 → 0 → 1 → …
- Each frame consists of 6 * 1 + 3 * 27 = 87 numbers.
  - The first 6 numbers are (XYZ translation, XYZ rotation) of the root.
  - Next, every 3 numbers are (XYZ rotation) of each joint.
  - The rotation order is YXZ i.e. $\hat{v} = R_Z R_X R_Y v$ for an arbitrary vector v.

radian으로 바꿔서 넣어야함

Mesh data

- The mesh data will be provided in inc/binary/player.h header file.
  - playerTexels: the square texture
  - playerSize: the resolution of playerTexels
  - playerVertices: the mesh
  - playerIndices: the index of the mesh
- Vertex structure is slightly modified for skinning.
  - Vertex.bone: the index of skinned skeleton
  - Vertex.weight: the weight of skinning

Vec 4 of

Problem

scene.cpp

- Write the code in Scene::update(float deltaTime) function.
  - Calculate the elapsed time from the start by accumulating deltaTime.
  - Repeat the animation every 4 seconds. 0->1->2->3 이 4초이내
  - Convert the animation from Euler angles to quaternions.
  - Interpolate the animation.
  - Update VBO and IBO of the object.
  - Apply the skinning with the weight blending.

```
void Scene::update(float deltaTime) {
    Scene::program->use();

    Scene::camera->update();

    ////////////////////////////
    /* TODO */

    // Line Drawer & Debugger
    glLineWidth( width: 20);
    Scene::lineDraw->load( vertices: {{ pos: vec3( x: -20.0f,  b: 0.0f,  c: 0.0f)}, { pos: vec3( x: 20.0f,  b: 0.0f,  c: 0.0f)}},  indices: {0, 1});
    Scene::lineDraw->draw();

    // LOG_PRINT_DEBUG("You can also debug variables with this function: %f", M_PI);

    ////////////////////////////

    Scene::player->load(playerVertices, playerIndices);
    Scene::player->draw();
}
```

Problem (optional)

scene.cpp

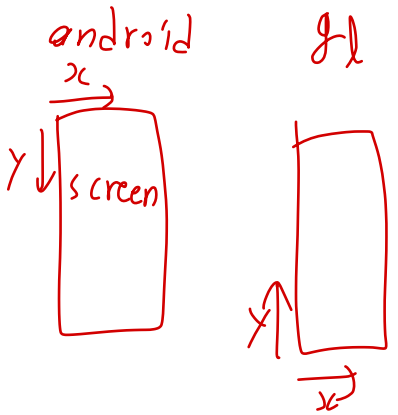
- Rotate the character only when dragging the mouse.
  - You may fill in the **mouseMoveEvents** and **mouseDownEvents**.
  - The parameters (x, y) are the screen space coordinates. 안드로이드에서의 x, y 좌표값 opengl이랑 다름
  - The above methods are called automatically, so there is no need to worry about how it works.

```
void Scene::mouseDownEvents(float x, float y) {
    ////////////////////////////
    /* TODO: Optional problem
     * object rotating
     * Automatically called when mouse down
     */

    ////////////////////////////
}

void Scene::mouseMoveEvents(float x, float y) {
    ////////////////////////////
    /* TODO: Optional problem
     * object rotating
     * Automatically called when mouse move
     */

    ////////////////////////////
}
```



Tip

- You can use the OpenGL Mathematics (GLM) functions.
  - GLM is a C++ header only library for graphics software based on the GLSL specifications.
  - GLM functions are included in “app/src/main/cpp/inc/glm”.
  - Useful glm function
    - `glm::rotate`
    - `glm::translate`
    - `glm::quat_cast`
    - `glm::mat4_cast`
    - `glm::mix`
    - `glm::slerp`
  - Documentations can be found here. [link]

Tip

- Visualize the skeleton.
  - The object with the line drawer will be provided.

```
// Line Drawer & Debugger
glLineWidth( width: 20);
Scene::lineDraw->load( vertices: {{ .pos: vec3( a: -20.0f,  b: 0.0f,  c: 0.0f)}, { .pos: vec3( a: 20.0f,  b: 0.0f,  c: 0.0f)}},  indices: {0, 1});
Scene::lineDraw->draw();

// LOG_PRINT_DEBUG("You can also debug variables with this function: %f", M_PI);
```

  - Fill the VBO and IBO to visualize the skeleton.

Tip

mesh가 너무 비정상적으로 크면 1/3로 scale down

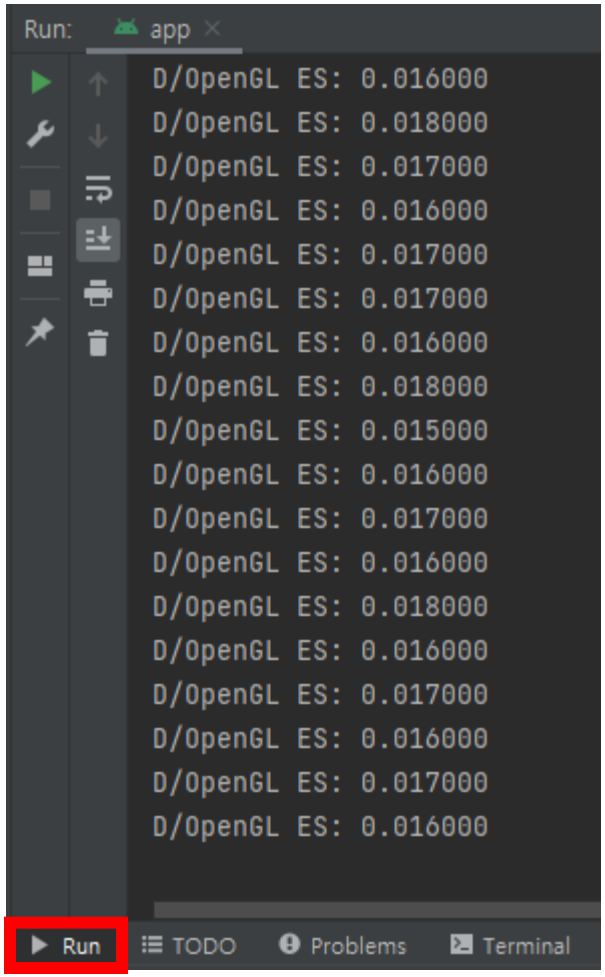
- Overlap the skeleton and the mesh.
  - If the skeleton and the mesh view the different direction, the result will be weird.
- Apply the skinning without the weight blending.
  - The result is good enough.
- The mesh and the skeleton are too big.
  - Note that the mesh is scaled down to 1/3.

Tip

- 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
  - May 25 (Wed) 14:00
- Submission files ({student_id}_{name}.zip)
  - app/src/main/cpp/src/scene.cpp
- Please follow the submission format!!
- Submission to Blackboard
- Contact
  - TA email: 2022.CG.TA@gmail.com

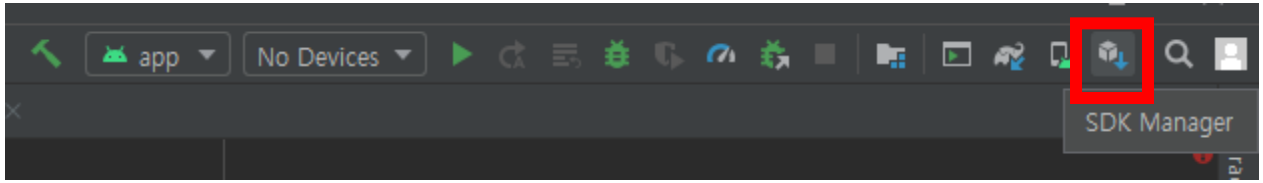
Appendix: Android Studio Setup Guide

Android Studio

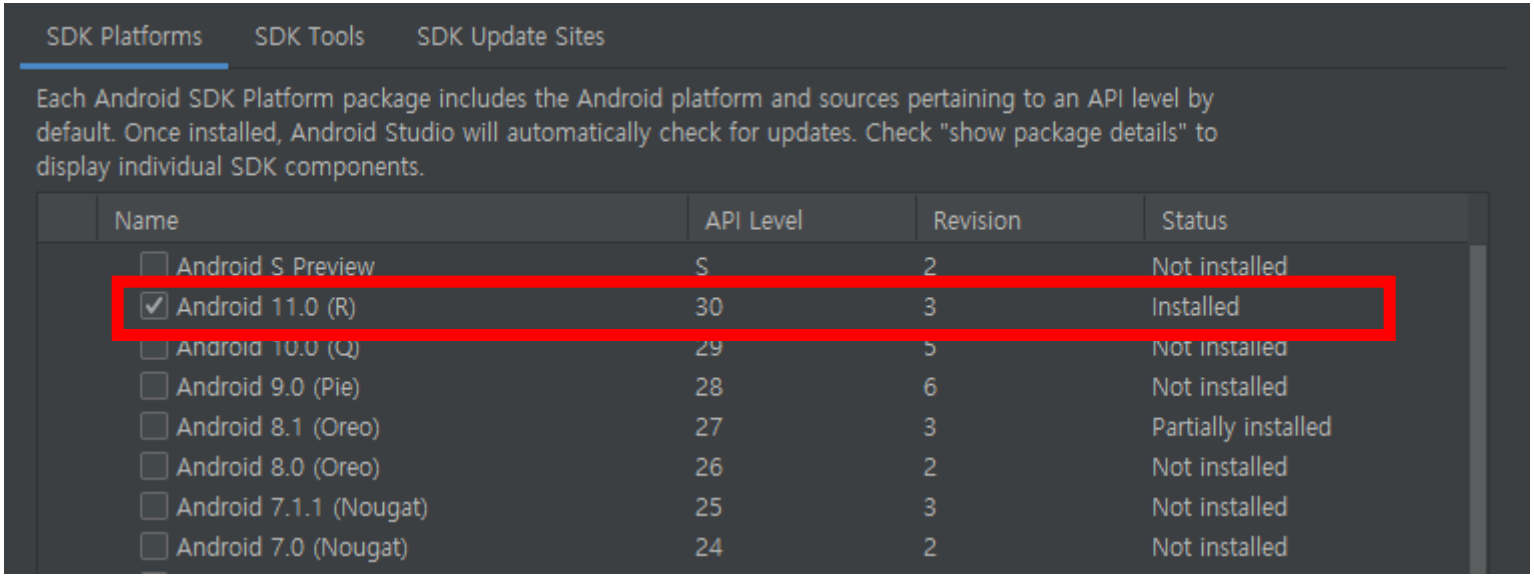
- **Android Studio** is the official integrated development environment (IDE) for the Android platform.
- Android Studio can be downloaded from the official website. [link]

Android SDK

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



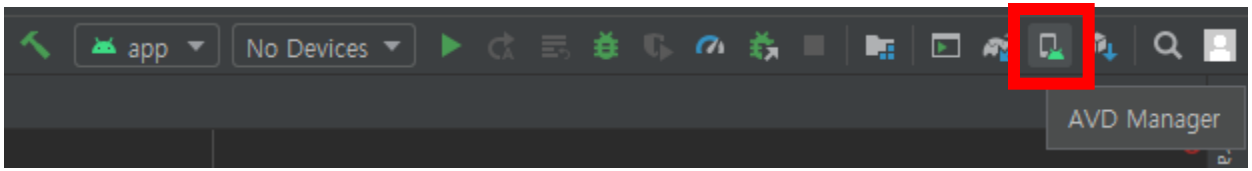
- The **latest platform** (API level 30) will be installed automatically.



Android SDK

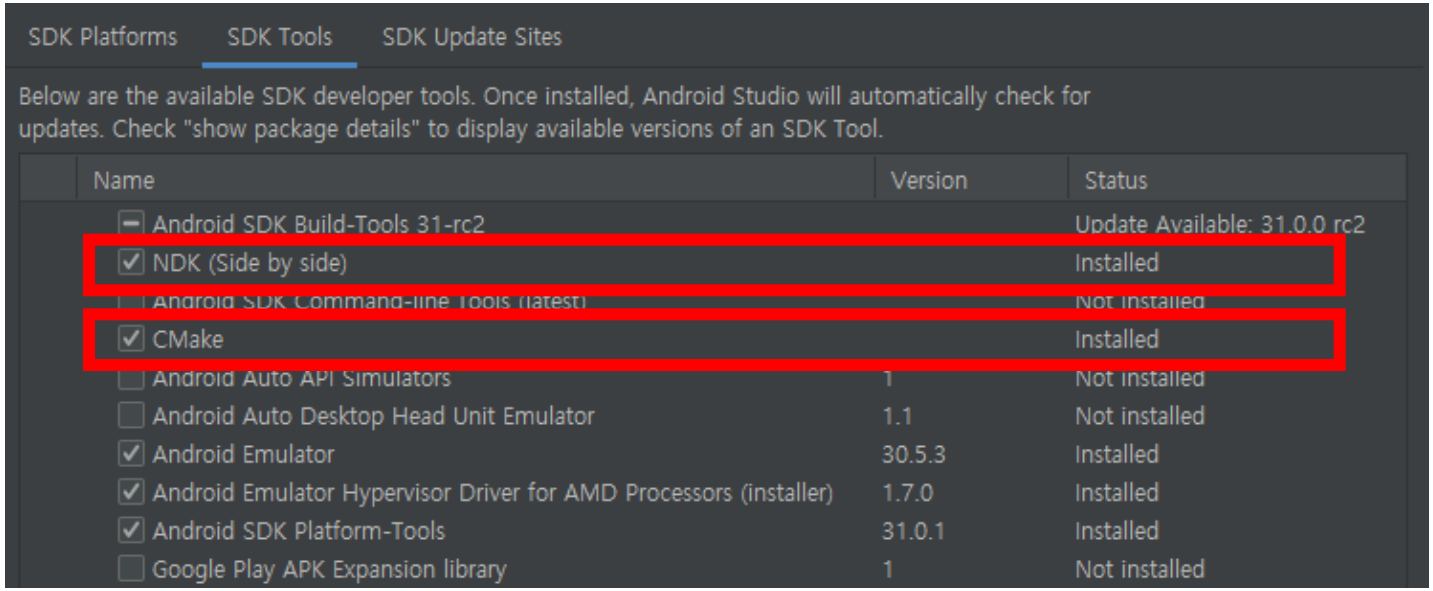
- You may need to install **additional platforms** to try USB debugging on your smartphone.

- 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.



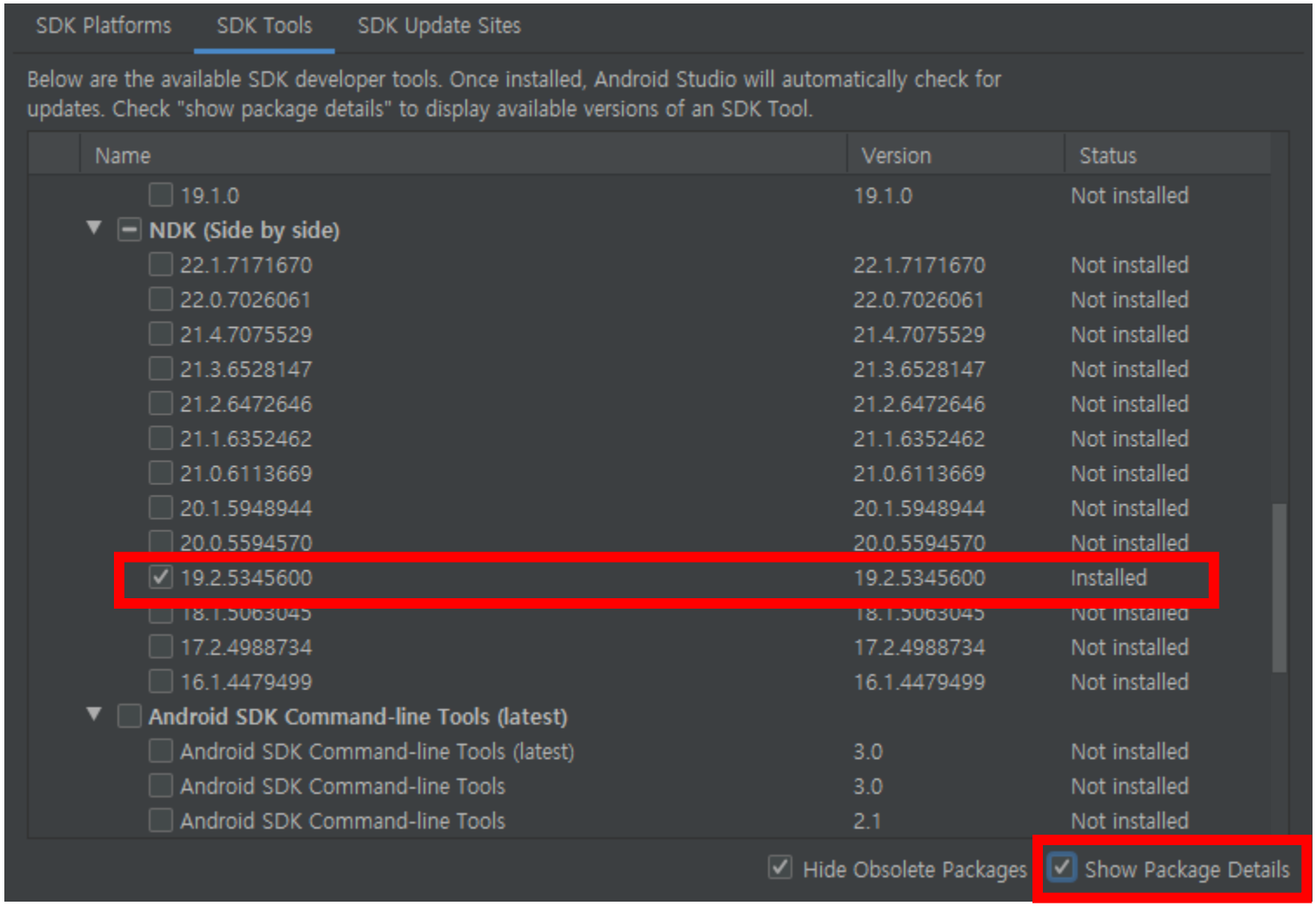
Android SDK

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



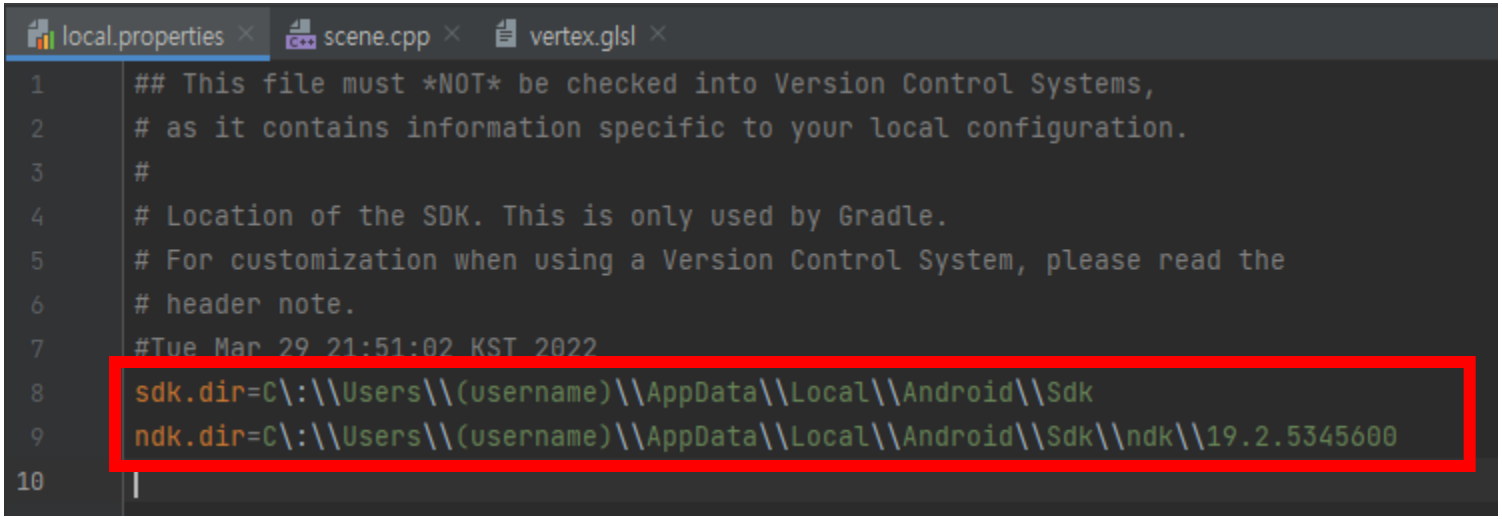
Android SDK

- 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