Homework 2

COSE331 Computer Graphics

Goal

- Implement normal mapping in shader
- Implement forward kinematics
- Apply skinning on the mesh
- Interpolate the animation between key frames

Initial state

- Character is in T-pose.

Final Result

- Character runs through the screen.
- When the Button is toggled, the character's moving state should be changed.

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
  - playerNormal : the normal map texture
- Vertex structure is slightly modified for skinning.
  - Vertex.bone: the index of skinned skeleton
  - Vertex.weight: the weight of skinning

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

Skeleton data

- Assume that the joints within the red box influence the movement of the Head.

```
const vector<string> jNames = {
        "Root",
        "Spine",
        "LThigh",
        "LCalf",
        "LFoot",
        "LToe0",
        "LToe0Nub",
        "RThigh",
        "RCalf",
        "RFoot",
        "RToe0",
        "RToe0Nub",
        "Spine1",
        "Neck",
        "Head",
        "HeadNub",
        "LClavicle",
        "LUpperArm",
        "LForearm",
        "LHand",
        "LFinger0",
        "LFinger0Nub",
        "RClavicle",
        "RUpperArm",
        "RForearm",
        "RHand",
        "RFinger0",
        "RFinger0Nub"
};
```

Animation data

- The animation data will be provided in
  - The animation has 4 key frames.
  - 0 → 1 → 2 → 3 → 0 → 1 → …
- Each frame consists of 6 * 1 + 3 * 27 =
  - The first 6 numbers are (XYZ translatio

Problem 1

- Write the Code in shader file.

- Apply normal mapping to the character.
  - The normal map image is stored in playerNormal in cpp/inc/binary/player.h file.
  - Alignment with Tangent Space is not required.

- [Extra point] will be awarded for implementing normal mapping with respect to Tangent Space.

Problem 2

- 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.
  - Convert the animation from Euler angles to quaternions.
  - Interpolate the animation.
  - Update VBO and IBO of the object.
  - Apply skinning to the vertex and normal using weight blending.

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

    /*
     *
     * Write your code.
     *
     */

    // Line Drawer
    // glLineWidth(20);
    // Scene::lineDraw->load({{vec3(-20.0f, 0.0f, 0.0f)}, {vec3(20.0f, 0.0f, 0.0f)}}, {0, 1});
    // Scene::lineDraw->draw();

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

Problem 2

- The flag variables are automatically set when the toggle button is pressed/released.
  - When the toggle button is pressed, the flag is set to false and stop the animation.
  - When the toggle button is released, the flag is set to true and move the animation.

- Ex) When the button is pressed, stop the animation of the head.
  - buttonFlag is set to false


```
void Scene::setButtonFlag(bool flag)
{
    Scene::buttonFlag = flag;
}
```

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
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();
```

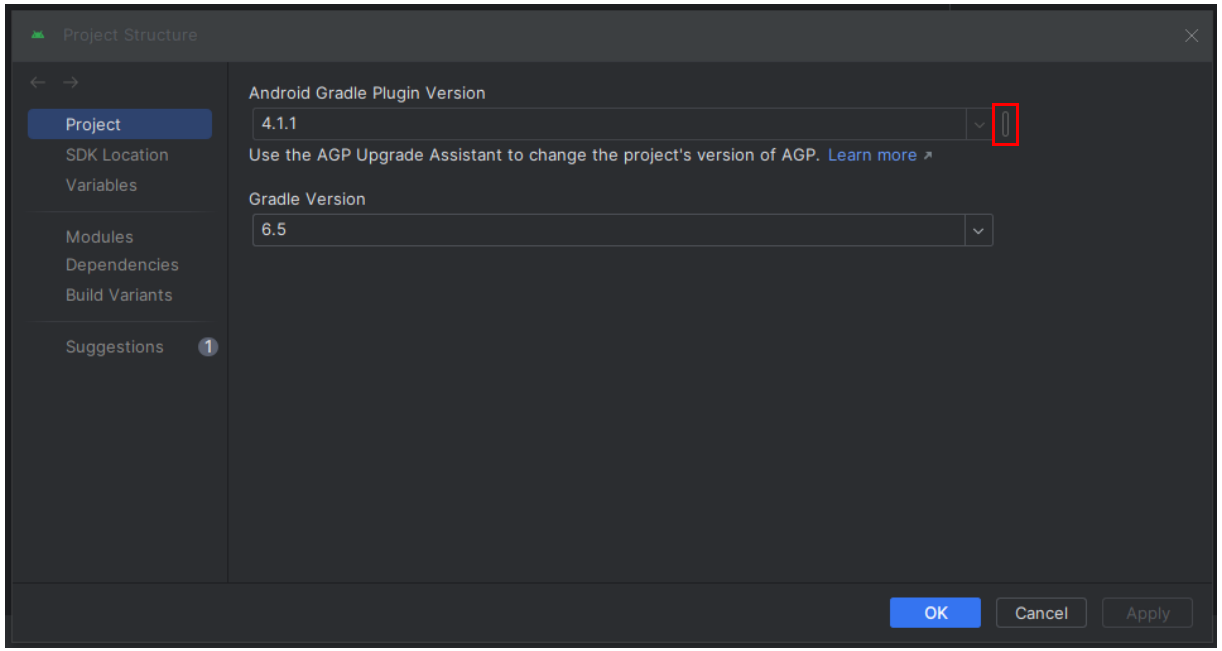
  - Fill the VBO and IBO to visualize the skeleton.

Tip

- 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.

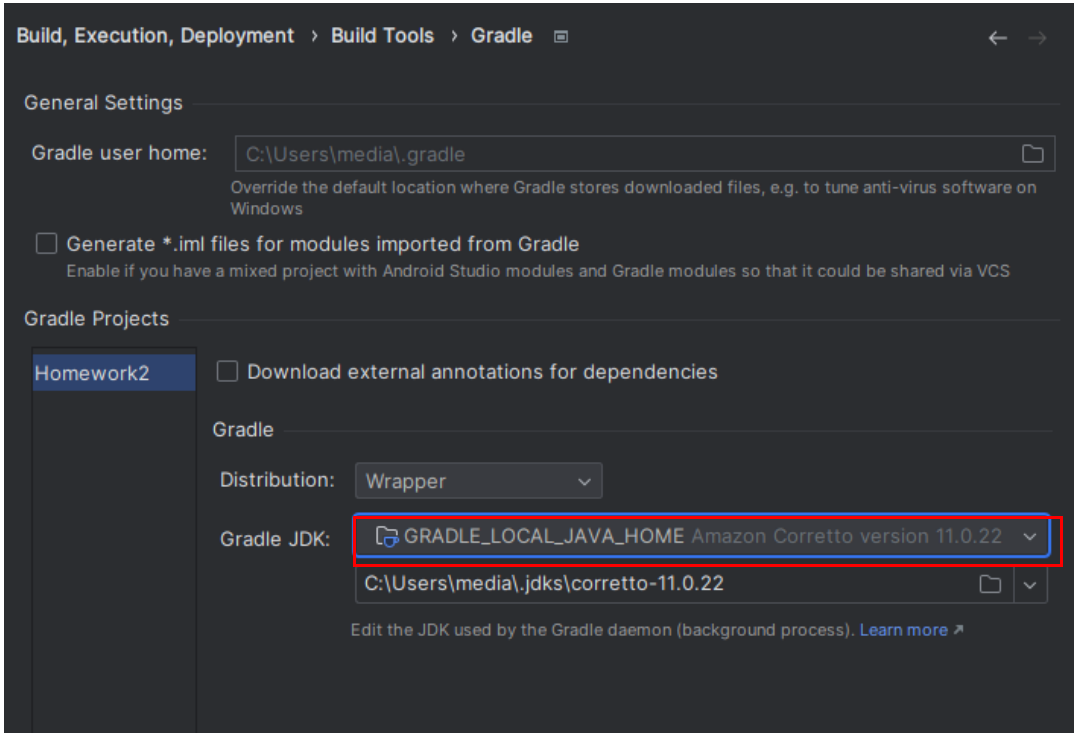
Tip

- There might be issues with the Android Studio Gradle and JDK settings.
  - To change the settings, go to “File > Project Structure”.
  - To change the Android Gradle Plugin version, click the small button on the side.
  - Please configure the Project Structure as shown in the image below.



Tip

- There might be issues with the Android Studio Gradle and JDK settings.
  - To change the settings, go to "Settings > Build, Execution, Deployment > Build Tools > Gradle".
  - Set the Gradle JDK to corretto-11.



Guidelines on Plagiarism

- No excuses for cheating.
  - Anyone who copies code from other students will fail this class.

- Submit **all the references** used in your code.
  - All online code references, including open source, blogs, and records from generative language models, **must be submitted beforehand**.
  - Once a case is judged to be plagiarism, any additional reference submissions will not be accepted as explanatory materials.

Submission

- Deadline
  - June 13 (Wed) 17:00
- Submission files ({student_id}_{name}.zip)
  - Scene class file (app/src/main/cpp/scene.cpp)
  - Vertex/fragment shader file (assets/vertex.glsl, assets/fragment.glsl)
  - code reference document (reference.pdf * any format will be fine)
- Submission to Blackboard
- Contact
  - TA email: 2024.cg.ta@gmail.com