Skeletal Animation

Instruction

Model Loading

- Collada file format (.dae)
  - Mesh
    - Vertices
      - Position
      - Normal
      - Texture coordinate
    - Indices
  - **Skeleton**
    - **Bone names**
    - **Hierarchy**
    - **Transformation matrix**
    - **The number of bones that influences the vertex**
    - **Weights**

https://learnopengl.com/Guest-Articles/2020/Skeletal-Animation

Model Loading

- Collada file format (.dae)
  - **Animation**
    - **keyframes**

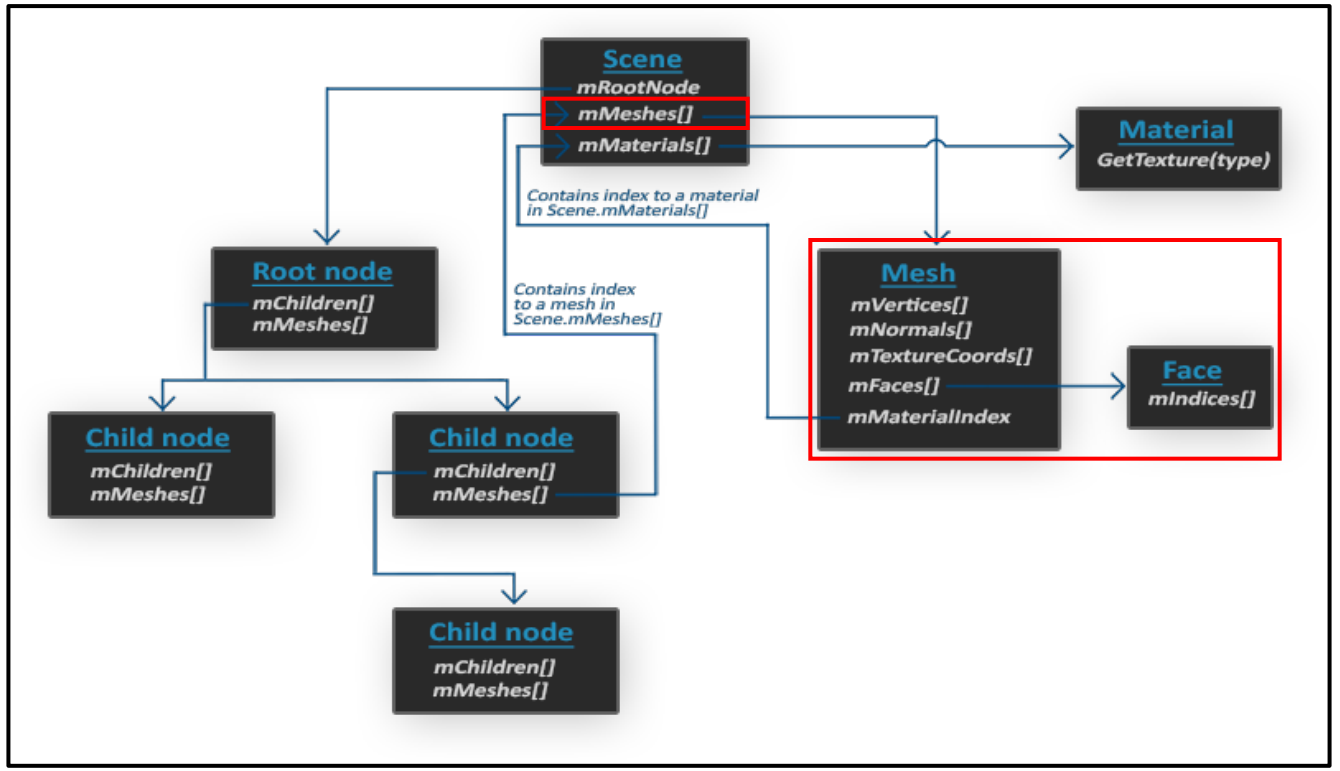
https://learnopengl.com/Guest-Articles/2020/Skeletal-Animation

Model Loading

- Assimp 5.0.1 (3D model importer)
  - https://github.com/assimp/assimp

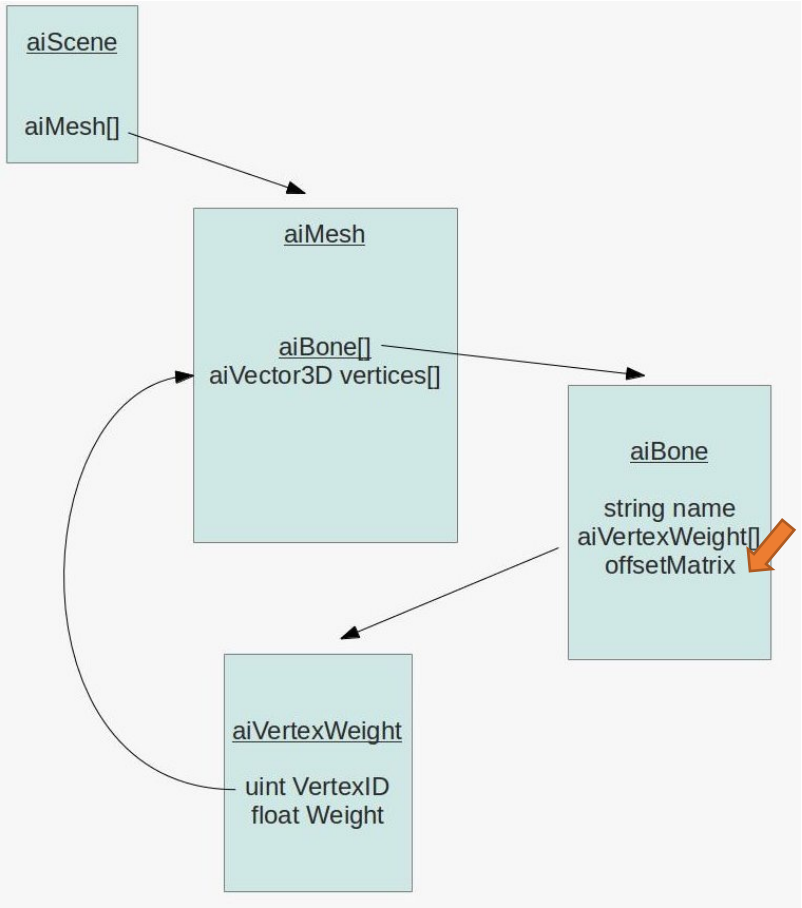
https://learnopengl.com/Guest-Articles/2020/Skeletal-Animation

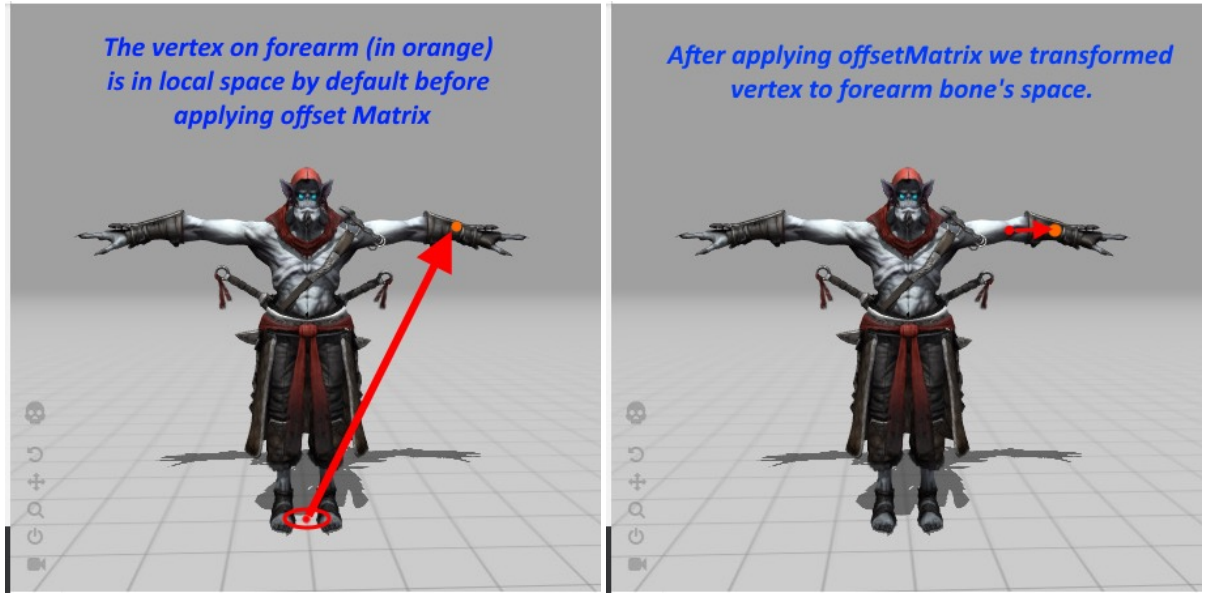
Mesh Data



https://learnopengl.com/Model-Loading/Assimp

Bone Data

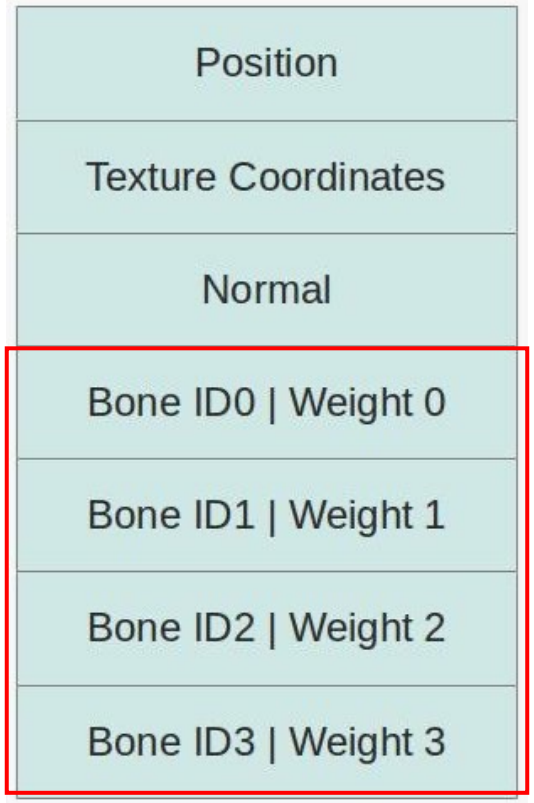




https://learnopengl.com/Guest-Articles/2020/Skeletal-Animation

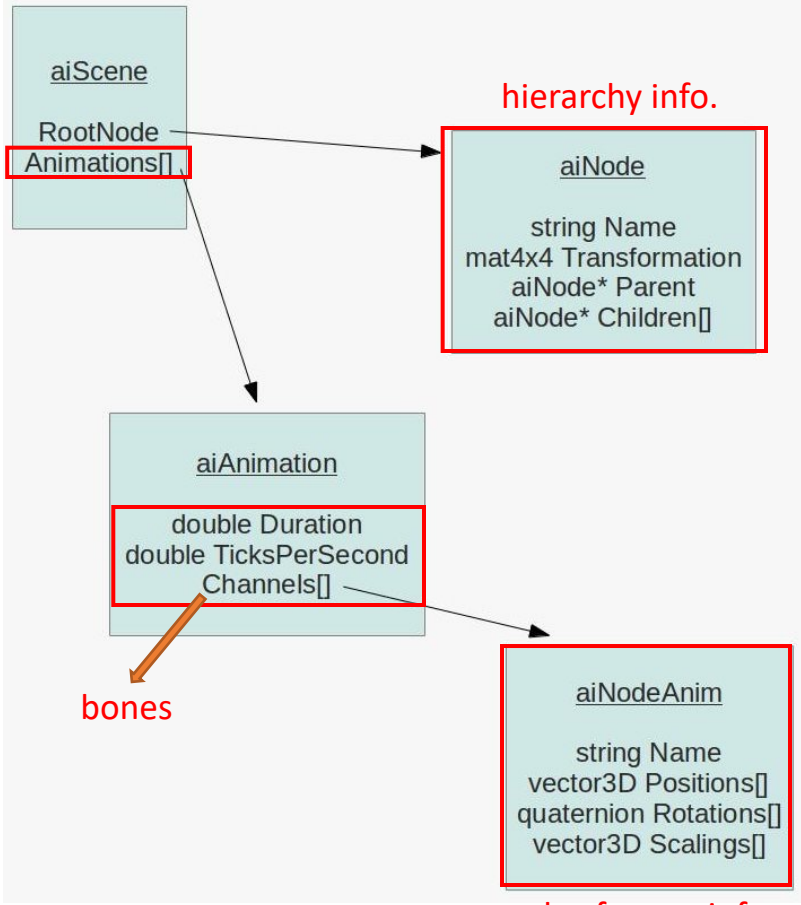
Bone Data

- Vertex Structure



https://learnopengl.com/Guest-Articles/2020/Skeletal-Animation

Animation Data



https://learnopengl.com/Guest-Articles/2020/Skeletal-Animation

Animation Data

Node hierarchy:

- **Scene**
  - Armature
    - bone_root
      - bone_body
      - …
  - Camera
  - Light

https://learnopengl.com/Guest-Articles/2020/Skeletal-Animation

Bone Transformation Calculation

- Calculate the bone transformations that go into the shader every frame

Current time

Output: Final Bone transformations

```
void boneTransform(double secs, vector<glm::mat4x4>& transforms)
{
    glm::mat4 identity_matrix(1.0f);

    double ticks = secs * ticks_per_second;
    float animation_time = 0;
    if (scene->HasAnimations()) {
        animation_time = fmod(ticks, scene->mAnimations[0]->mDuration);
    }

    if (run_animation) {
        readNodeHierarchy(animation_time, scene->mRootNode, identity_matrix);
    } else {
        readNodeHierarchy(0, scene->mRootNode, identity_matrix);
    }

    transforms.resize(num_bones);

    for (int i = 0; i < num_bones; i++) {
        transforms[i] = bone_matrices[i].final_transform;
    }
}
```

Process the node hierarchy

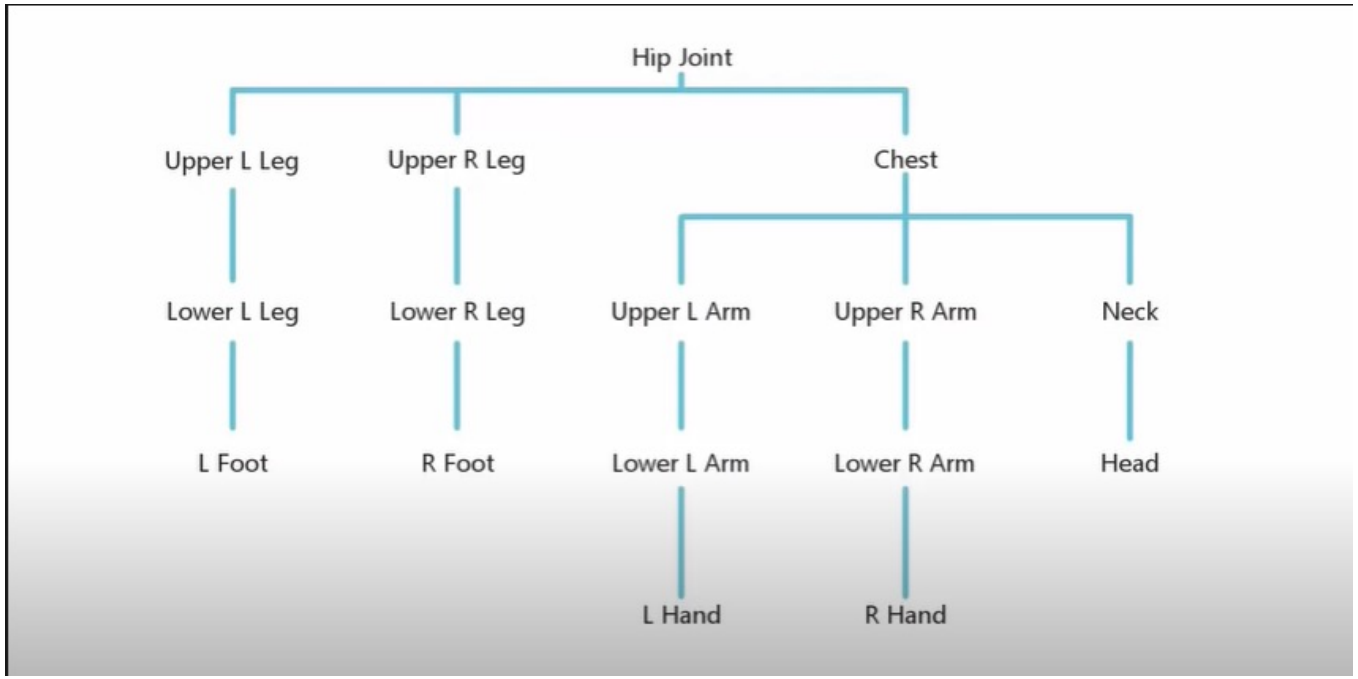
https://ogldev.org/www/tutorial38/tutorial38.html

Bone Transformation Calculation

- Process the node hierarchy and get the **final transformation** for **each bone**.
  1. Search for the node name in the **channels** of the animation.
  2. Interpolate the **scaling** vector, **rotation** quaternion, and **translation** vector
  3. **globalTransformation** = **parentMatrix** * interpolatedTransformation
  4. **finalTransformation** = globalTransformation * **boneInfo[nodeName].offset**

https://ogldev.org/www/tutorial38/tutorial38.html

Bone Transformation Calculation



https://ogldev.org/www/tutorial38/tutorial38.html

Skinning

- Linear blending skinning (in vertex shader)

```
mat4 bone_transform = bones[bone_ids[0]] * weights[0];
bone_transform += bones[bone_ids[1]] * weights[1];
bone_transform += bones[bone_ids[2]] * weights[2];
bone_transform += bones[bone_ids[3]] * weights[3];
```

https://ogldev.org/www/tutorial38/tutorial38.html

References

References

- 3D models
  - https://www.cgtrader.com/free-3d-models?file_types%5B%5D=25
- Skeletal Animation
  - https://learnopengl.com/Guest-Articles/2020/Skeletal-Animation
  - https://ogldev.org/www/tutorial38/tutorial38.html
  - https://www.youtube.com/watch?v=f3Cr8Yx3GGA&list=PLRIWtICgwaX2tKWCxdeB7Wv_rTET9JtWW
- Assimp (3D model importer)
  - https://assimp.org/index.php
- Blender
  - https://www.blender.org/