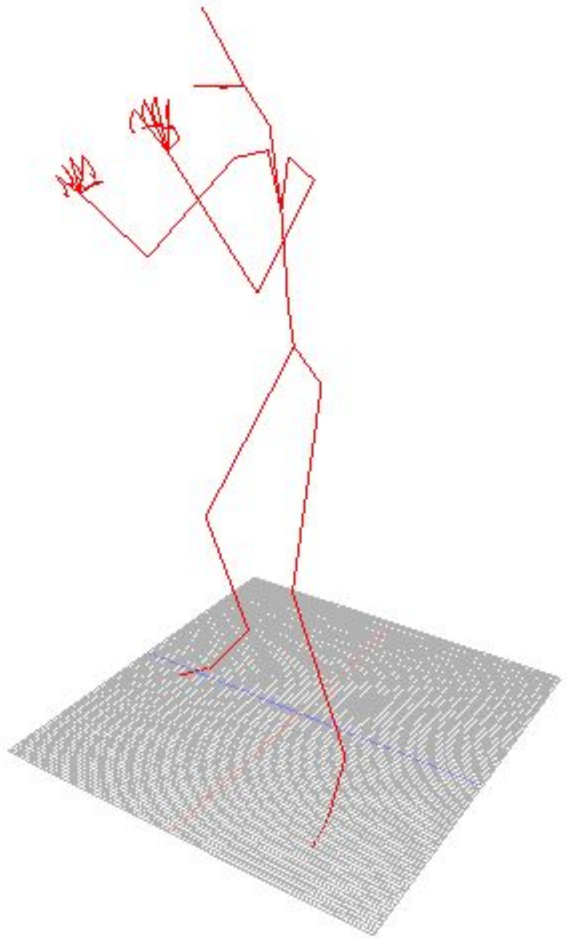
MVD: Advanced Graphics 2

22 - Skinned Animation

alunthomas.evans@salle.url.edu

laSalle ENG
Universitat Ramon Llull

Check wireframe animation



Skinned Mesh multiplication order

Raw Vertex Position

Skin Global Bind Matrix
(optional ‘skin model matrix’)

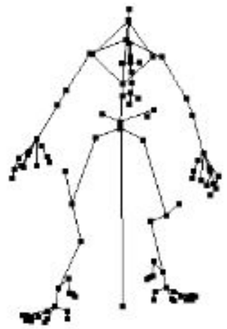
For each joint:
(that affects this vertex)

Joint Bind Matrix
(transform to bind-pose)

Joint transform
(bind-pose to current keyframe)

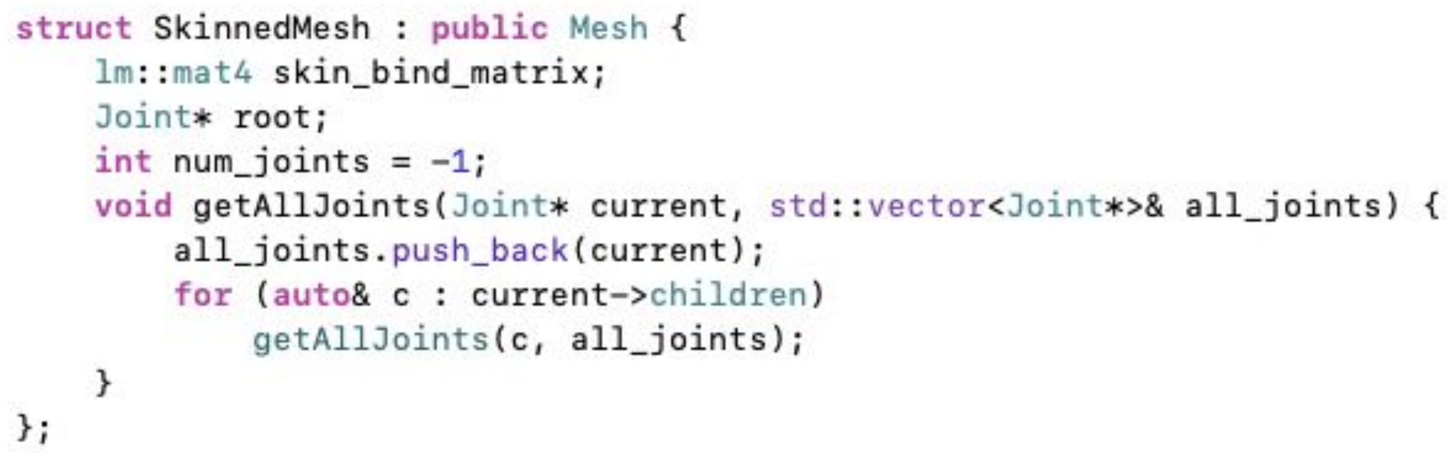
Joint weight
(for this vertex)

Final Vertex Position



SkinnedMesh Component

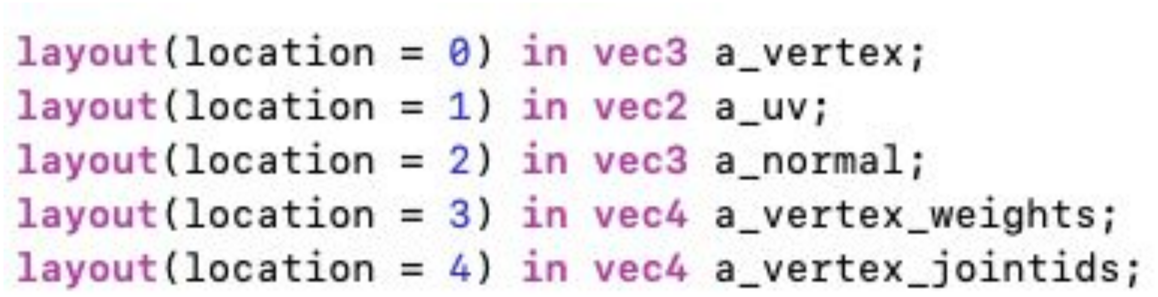
Exactly the same as the ‘Joint Chain’ from previous class, but with one extra variable: the skin_bind_matrix (moves mesh globally into bind pose)



```
struct SkinnedMesh : public Mesh {
    lm::mat4 skin_bind_matrix;
    Joint* root;
    int num_joints = -1;
    void getAllJoints(Joint* current, std::vector<Joint*>& all_joints) {
        all_joints.push_back(current);
        for (auto& c : current->children)
            getAllJoints(c, all_joints);
    }
};
```

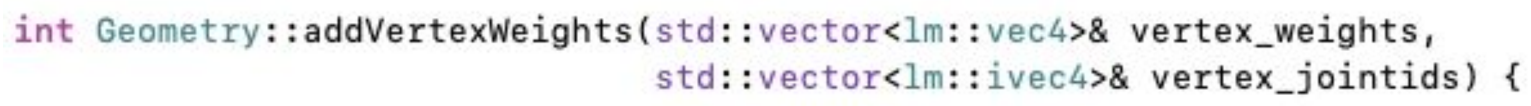
Joint shader (phong-anim.vert)

Task: Add attributes



```
layout(location = 0) in vec3 a_vertex;
layout(location = 1) in vec2 a_uv;
layout(location = 2) in vec3 a_normal;
layout(location = 3) in vec4 a_vertex_weights;
layout(location = 4) in vec4 a_vertex_jointids;
```

TASK: Add vertex weights to Geometry



```
int Geometry::addVertexWeights(std::vector<lm::vec4>& vertex_weights,
                               std::vector<lm::ivec4>& vertex_jointids) {
```

Collada parsers calls this function and provides it with vector of vec4s (weights, and vector of ivec4 (joint ids)

Need to convert both of these to single float arrays

Then upload to GPU; bind VAO and use glBufferData

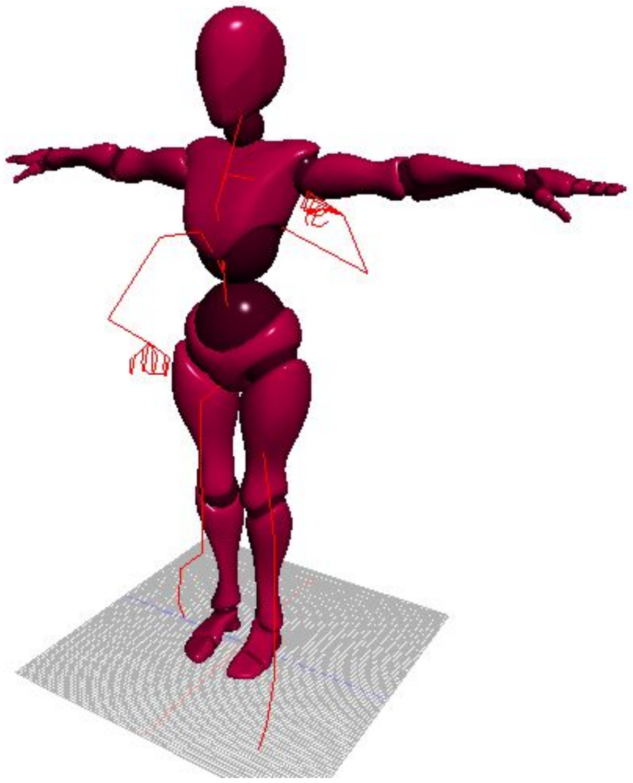
Drawing the unskinned mesh

Implement function renderSkinnedMeshComponent_

For now, simply 'wrap' renderMeshComponent_

U_VP from camera view projection

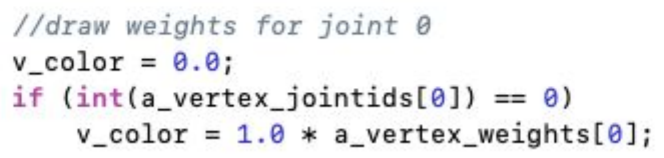
We will add extra uniforms later



```
auto& skinnedmesh_components = ECS.getAllComponents<SkinnedMesh>();
for (auto &skinnedmesh : skinnedmesh_components) {
    checkShaderAndMaterial_(skinnedmesh);
    renderSkinnedMeshComponent_(skinnedmesh);
}
```

Visualising the weights

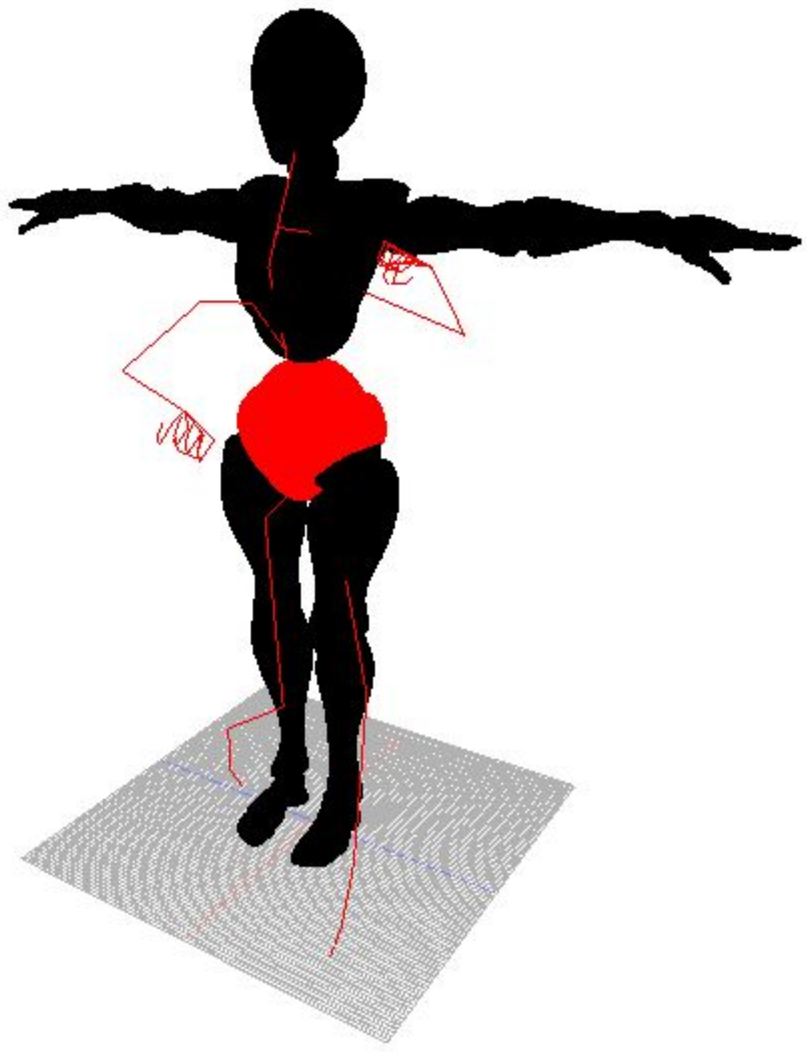
Add out varying to vertex shader



```
//draw weights for joint 0
v_color = 0.0;
if (int(a_vertex_jointids[0]) == 0)
    v_color = 1.0 * a_vertex_weights[0];
```

draw this color in fragment shader

note converting float back to int



Task: Add shader uniforms

For shader uniforms, we need

- matrix to move skin to bind pose
- array of joint bind-pose matrices (move joint into bind pose)
- array of joint animation matrices (change per frame, move from bind-pose to animation
- View_Projection matrix (instead of MVP)

Task: upload shader uniforms in renderSkinnedMesh_()

Recursive function to obtain:

- joint bind-pose matrices
- joint animation matrices

same depth-first traversal as for bones - use existing getJointMatrices function

upload uniforms as for bones

Skin bind matrix => SkinnedMesh.skin_bind_matrix

The shader equation

Step one:

multiply raw vertex position by the skin global bind matrix.

Leave this as a new constant

Raw Vertex Position

Skin Global Bind Matrix
(optional ‘skin model matrix’)

Joint Bind Matrix
(transform to bind-pose)

Joint transform
(bind-pose to current keyframe)

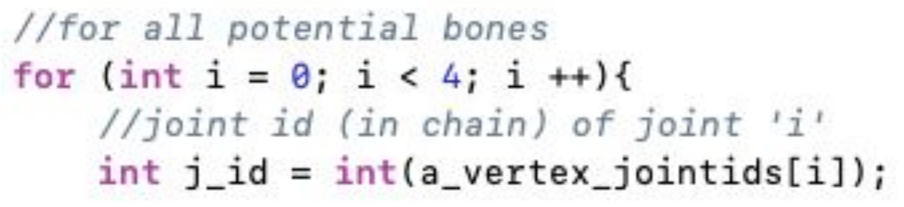
Joint weight
(for this vertex)

Final Vertex Position

The shader equation

Step two:

Loop all four potential joints for each vertex, and get joint id for each one:



```
//for all potential bones
for (int i = 0; i < 4; i ++){
    //joint id (in chain) of joint 'i'
    int j_id = int(a_vertex_jointids[i]);
```

Raw Vertex Position

Skin Global Bind Matrix
(optional ‘skin model matrix’)

Joint Bind Matrix
(transform to bind-pose)

Joint transform
(bind-pose to current keyframe)

Joint weight
(for this vertex)

Final Vertex Position

The shader equation

Step three:

The equation inside the loop:

final vert +=
    weight for this index *
    position_matrix[joint id] *
    bind matrix[joint id]
    vertex_bind_pose

Raw Vertex Position

Skin Global Bind Matrix
(optional 'skin model matrix')

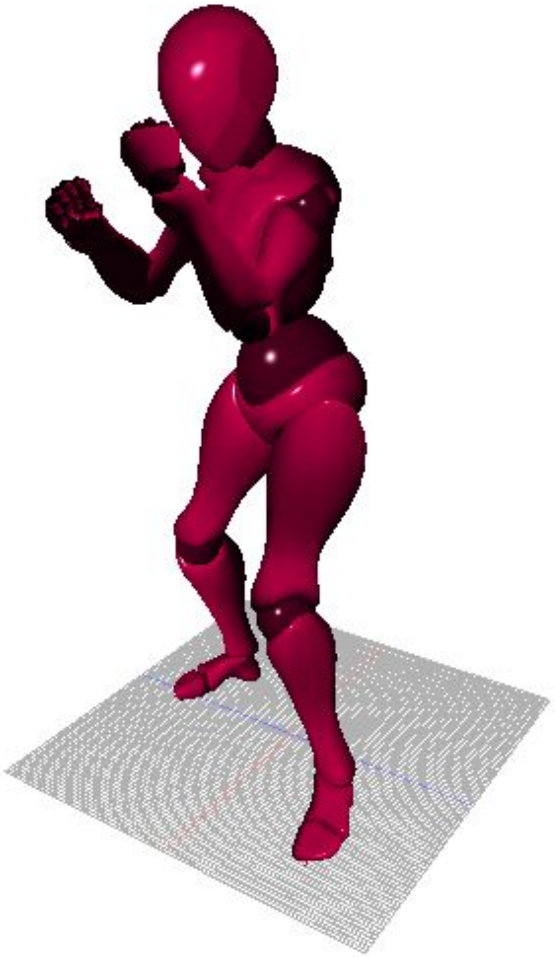
Joint Bind Matrix
(transform to bind-pose)

Joint transform
(bind-pose to current keyframe)

Joint weight
(for this vertex)

Final Vertex Position

Result



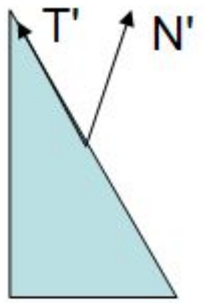
Normals

Normal matrix revision

A normal is a direction so the normal matrix is 3x3 instead of 4x4

technically we should also take the inverse transpose, as this removes any scale component.

But this is only relevant if we scale anisotropically



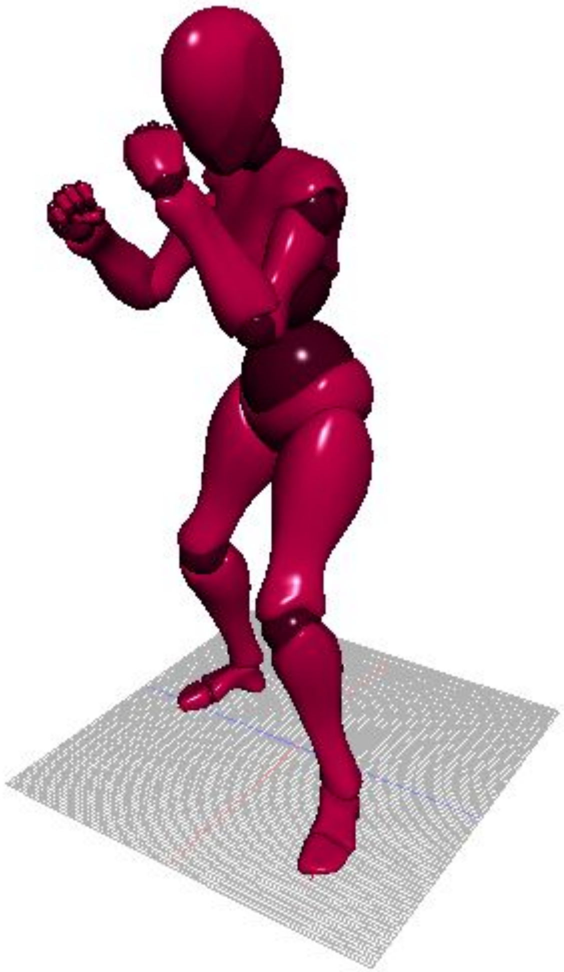
Task

Use mat3 glsl function to make a 3x3 copy of all matrices

Do same multiplication, but use a_normal as input

output to final_normal

Result



Finally:animate

In animation system,

When tick, call incrementJointFrame_ on root joint of skinned mesh