Freestyle Project: Animation and Skinning:

Einleitung

Ich habe bisher einige Vorlesungen über Computergrafik gehört. Leider habe ich bisher nur am Rande von Animationen gehört. Jedoch habe ich bisher noch nie irgendwas in die Richtung implementiert, und deshalb sehe ich das als Gelegenheit darüber mehr über Animationen zu lernen.

Das Ziel ist es eine animierte Figur zu laden und zu rendern.

Rendering Techniken

In meiner Abgabe soll eine Figur mit einem zugehörigen Skelet gerendert werden. Die Figur wird mithilfe von Sinnig gerendert, d.h. die gewichteten Oberflächen Vertices werden aufgrund der Positionen der zughörigen Bones an ihre Position gebracht.

Anschließend wird das Skelett mittels Keyframes und Interpolation zwischen den Frames animiert.

Implementation

Initial wird ein Mesh mit der assimp Bibliothek geladen (https://github.com/assimp/assimp). Die Bibliothek unterstützt einige Formate. Die Bibliothek liest auch Bones und Animationen, wenn sie vorhanden sind und transferiert diese in ein einheitliches Format.

Das Skelet hat eine Baumstruktur und jeder Bone kennet seinen „Parent“. Ein Bone besteht aus einer Skalierung, einet Rotation und einer Translation. Durch die Baumstruktur und der relativen Transformationen jeder Bone lässt sich so die absolute Transfomation errechnen.

$$BoneTransform_{Child} = BoneTransform_{Parent} * (Translate * Rot * Scale)_{Child}$$

Das Mesh von assimp ist nicht in Bonekoordinaten sondern in Objektkoordinaten. Deshalb muss mit die Offset Matrix des entsprechenden Bones an die Transformationsmatrix multipliziert werden. Nach der Transformation in der Bone Koordinaten muss das Mesh wieder zurück in Weltkoordinaten transformiert werden. Alle Bone Transformationen werden als Matrix Array an den Vertex-Shader übergeben.

FinalBoneTransform = InverseWorldTransform * BoneTransform * BoneOffset

Für das animieren eines Meshes mit Bones brauch jedes Vertex noch informationen über die Indizes der zugehörigen Knochen und ein Gewicht wie viel Einfluss der Knochen hat. In assimp werden diese Informationen im aiBone gespeichert. Initial müssen diese Werte den jeweiligen Vertex zugeordnet werden und in einem weitern Vertex Array abgelegt werden. Dieser wird zusätzlich zur Position, Normale etc. an den Vertex Shader übergeben, sodass sie da zur Verfügung stehen.

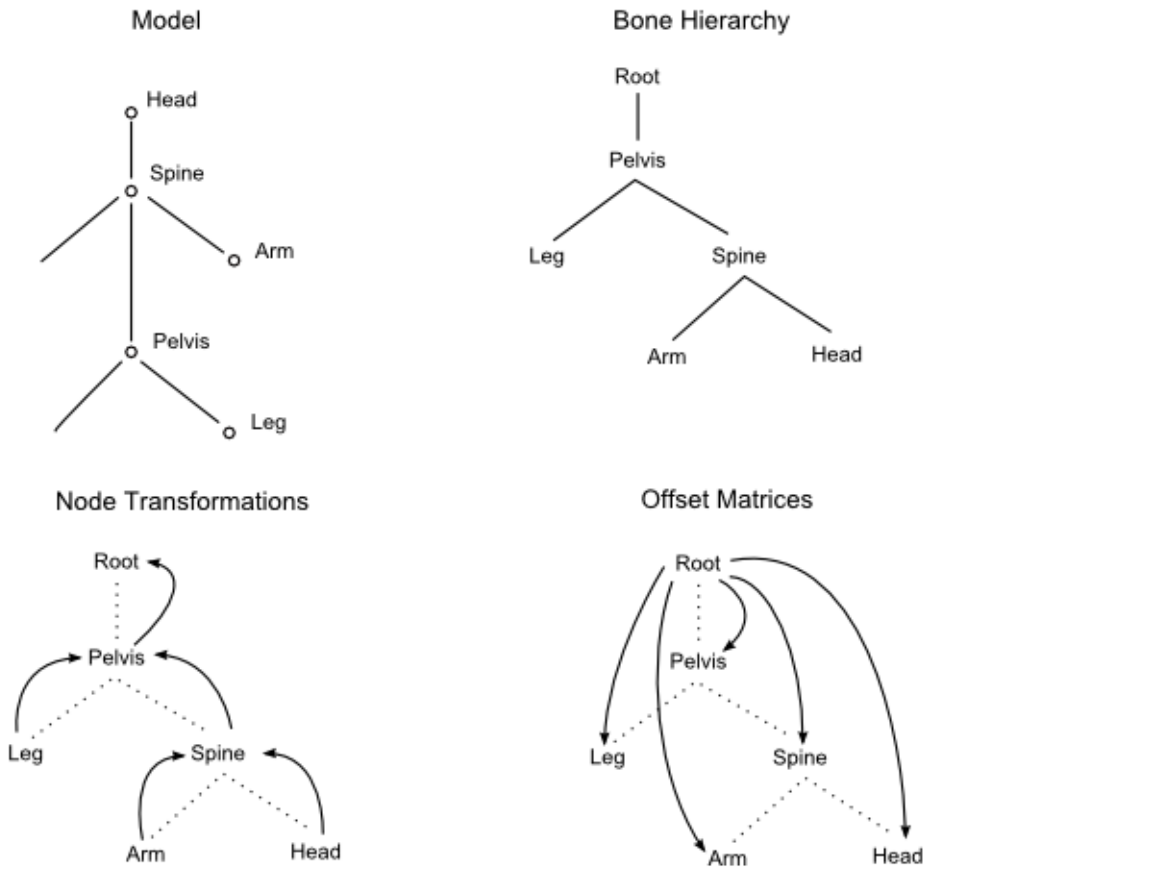
```
layout (location = 3) in ivec4 BoneIDs; //4 Id’s
layout (location = 4) in vec4 Weights; //4 Gewichte
```

Das Skinnig passiert im Vertex-Shader. Aus den Bone Array, dem Bone Index des Vertex und den Gewichten des Vertex wird die Weltkoordinate errechnet:

```
mat4 BoneTransform = Bones[BoneIDs[0]] * Weights[0];
   BoneTransform += Bones[BoneIDs[1]] * Weights[1];
```

```
    BoneTransform += Bones[BoneIDs[2]] * Weights[2];
    BoneTransform += Bones[BoneIDs[3]] * Weights[3];

vec4 worldPos = BoneTransform * vec4(inPosition,1.0);
```



$$FinalTransform_{Head} = Offset_{Head} \times (Scale \times Rot \times Trans)_{Head} \times (Scale \times Rot \times Trans)_{Spine} \times (Scale \times Rot \times Trans)_{Pelvis}$$

Assimp lädt, wenn vorhanden auch die Animationen in eine aiAnimation. Jede Animation hat eine Dauer und ein Array of Channels. Jeder Channel ist einem Bone zugeordnet und enthält alle Transformationen dieses Bones während der Animation. Zu einem Zeitpunkt der Animation muss man das Keyframe davor und danach also die vorherige und die nachfolgende Transformation heraussuchen. Aus deine beiden Transformationen wird durch lineare Interpolarisation die aktuelle Transformation errechnet. Als Faktor wird der Zeitpunkt zwischen den beiden Frame Zeiten genommen. Die Transformation ist getrennt in Translation Sklierung und Rotation gespeichert. Die Rotation ist als Quaternion gespeichert und für diese bietet Assimp eine Interpolier Methode. Für Translation und Skalierung soll durch Centripetal Catmull-Rom Spliens interpoliert werden. Für die Interpolarisation werden 4 Punkte (P0-4) benötig um zwischen den Punkten P1 & 2 ein Spline zu berechnen. Jedem Punkt wird eine Zeit t_i zugeordnet: $t_0 = 0$ & $t_{i+1} = |P_{i+1} - P_i|^{0.5} + t_i$
Mit den Zeiten und den Punkten wird mit der Barray and Goldman Pyramide die Interpolierte Position errechnet (https://en.wikipedia.org/wiki/Centripetal_Catmull%E2%80%93Rom_spline)

Am Anfang und am Ende der Animation werden die durch mPostState und mPreState definierten Verhalten angewendet. Z.B. Linear bedeutete das Lineare Interpolieren am Ende.

Anschließend werden aus den interpolierten Werten Matrizen erstellt die die Bone Transformation ersetzten. Die FinalBoneTransform wird daraus wie oben beschreiben errechnet.

Grading

Laden eines Mesh, der Bones und Animationen mit assimp (1)

Korrektes ausrechnen der absoluten Transformationen der Bones (4)

Rendern des Mesh mit Bones (Berechnen der Vertex Positionen im Shader) (2)

Interpolation zwischen 2 Keyframes mit Catmull-Rom Splines (6)

Animation des Skelets und des Mesh (2)

Weitere Links

http://content.gpwiki.org/index.php/OpenGL:Tutorials:Basic_Bones_System

https://www.opengl.org/wiki/Skeletal_Animation

http://ogldev.atspace.co.uk/www/tutorial38/tutorial38.html