4 Skinning

Das animierbare Skelett aus Aufgabe 2/3 und das Modell aus Aufgabe 1 sollen nun zu einem animierbaren Avatar zusammengeführt werden. Dazu muss das Skelett in das Modell eingepasst und die Bewegungen der Knochen auf das Modell übertragen werden.

4.1 Pinocchio-Beschreibung erzeugen (3 Punkte)

Das Einpassen des Skeletts wird vom Tool Pinocchio übernommen, welches unter folgender Adresse erhältlich ist: http://www.mit.edu/ ibaran/autorig/pinocchio.html Das Tool kann als Konsolenanwendung verwendet werden und erzeugt für ein gegebenes Skelett und ein Modell ein angepasstes Skelett mit zugehörigen Knochengewichten. Daher muss als erstes das geladene ASF-Skelett in das Pinocchio-Format überführt werden. Implementieren Sie dazu die Funktion `write_pinocchio_file()` in `Skeleton.cpp`.

Bei Pinocchio-Dateien handelt es sich um einfache Textdateien, bei denen jede Zeile einem Gelenk im Skelett entspricht. In der Zeile stehen nacheinander die Knoten-ID, die x, y und z-Koordinaten der Knotenposition im globalen Koordinatensystem und die ID des vorhergehenden Gelenks in der Skeletthierarchie.

Bei der Erzeugung der Datei sind einige Punkte zu beachten. In dem Skelett dürfen keine degenerierten Knochen mit der Länge 0 enthalten sein. In ASF-Skeletten ist das typischerweise für den Wurzelknochen der Fall. Entwickeln Sie eine Strategie, wie sich solche Situationen vermeiden lassen ohne die Topologie zu verändern. Außerdem muss das Skelett so skaliert werden, dass es in den achsenausgerichteten Würfel zwischen $(0, 0, 0)^T$ und $(1, 1, 1)^T$ passt. Schließlich müssen alle Knoten, also auch Knochenenden von bspw. Händen und Füßen, enthalten sein.

Die Funktion wird über die entsprechenden Knöpfe im SkeletonViewer aufgerufen.

4.2 Skeletteinpassung (1 Punkt)

Rufen Sie das Pinocchio-Programm `DemoUI` mit Ihrem Modell und dem erstellten Skelett auf und speichern Sie sich das adaptierte Skelett `skeleton.out` und die generierten Knochengewichte `attachment.out` ab.

Sollte Pinocchio Ihr Eingabemodell nicht verarbeiten können (z.B. weil nicht-mannigfaltige Kanten enthalten sind), nutzen Sie ein anderes Modell. In Opal ist das Modell `spiderman.obj` enthalten, das zusammen mit `jump.asf` gute Ergebnisse liefert.

4.3 Adaption des Skeletts (4 Punkte)

Implementieren Sie die Funktion `read_pinocchio_file()` in `Skeleton.cpp`, die ein Pinocchio-Skelett lesen und das ASF-Skelett dahingehend anpassen soll. Beachten Sie, dass die Gelenke evtl. umnummeriert wurden, wenn diese in der Eingabedatei nicht fortlaufend nummeriert waren. Wurde das Eingabeskelett in einer depth-first Traversierung erstellt, bleibt die Reihenfolge der Knochen jedoch erhalten.

Passen Sie zu jedem Knochen die Richtung und Länge an (2 Punkte). Außerdem muss die Orientierungsmatrix vom globalen in das lokale Knochensystem angepasst werden. Dies soll so geschehen, dass die Knochenrichtung im lokalen System des Knochens unverändert bleibt (1 Punkt). Sie können dafür die Methode `Bone::add_axis_rotation()` nutzen, die eine weitere Matrix an den Beginn der Orientierungsmatrix multipliziert. Achten Sie darauf, dass die Position des Skeletts angepasst werden muss (1 Punkt). Aktualisieren Sie außerdem die Bounding-Box mit den Funktionen `reset_bounding_box()` und `add_point()`.

4.4 Zusatz: Erhalten der Freiheitsgradbegrenzungen (4 Zusatzpunkte)

Bei der Adaption des Skeletts wie oben beschrieben wird davon ausgegangen, dass alle Freiheitsgrade in der 0-Position und die Grenzen erhalten bleiben. Das ist sinnvoll, wenn nur kleine Korrekturen notwendig sind. Größere Abweichungen werden aber in der Regel durch Änderung der Freiheitsgradwinkel hervorgerufen (z.B. durch Beugen des Ellenbogens). Adaptieren Sie das Skelett durch Änderung der Freiheitsgradwinkel, sodass deren Grenzen valide bleiben. Nutzen Sie dazu ein numerisches Optimierungsverfahren. Die Adaption kann lokal für jeden Knochen einzeln geschehen (+2 Punkte) oder global für alle Knochen gleichzeitig (+2 Punkte). Letzteres hat den Vorteil, dass auch Rotationen um die Knochenrichtung beachtet werden können, die sich in der Regel nur auf nachfolgende Gelenkpositionen auswirkt. Eine solche Optimierung ermöglicht zum Beispiel das korrekte Abspielen einer Animation mit bereits angepasstem Skelett.

4.5 Knochengewichte laden (3 Punkte)

Aktivieren Sie in `main.cpp` die Registrierung des `SkinnedMeshViewer`. Dieser stellt ein Rendering für ein animiertes Oberflächenmodell bereit. In der Klasse `Mesh` ist ein einfachen OBJ-Reader implementiert, der Positions- und Facettendaten liest. Beachten Sie, dass der OBJ-Reader sehr einfach gehalten ist und nur Daten im angegebenen Format lesen kann. Sollte Ihre Datei in einem anderen Format vorliegen, kann diese bspw. mittels MeshLab umgewandelt werden. Entfernen Sie dazu beim Exportieren alle Haken für die einzelnen Attribute.

Mit den entsprechenden Buttons im SkinnedMeshViewer kann ein Modell geladen und angezeigt werden.

Implementieren Sie die Funktion `Mesh::read_attachment()`, die die von Pinocchio erzeugten Knochengewichte (`attachment.out`) laden kann. Jede Zeile dieser Datei enthält die Gewichte für alle Gelenke außer den Wurzelknoten. Suchen Sie die vier größten Werte heraus und normieren Sie die Summe dieser Gewichte auf 1.

4.6 Skinning (5 Punkte)

Als letzter Schritt sollen die Bewegungen der Knochen auf das Oberflächenmodell übertragen werden. Dazu müssen als erstes die Skinning-Matrizen für jeden Knochen berechnet werden. Diese Matrizen geben eine Transformation von der Binding-Pose des Knochens zur deformierten Pose an. Berechnen Sie diese Matrizen in `get_skinning_matrices()` der Klasse `Skeleton` (3 Punkte). Berechnen Sie dazu zusätzlich die Matrix `transformLocalToGlobal` in `Bone::calculate_matrices()` (1 Punkt). Beachten Sie die globale Verschiebung des Skeletts.

Erweitern Sie anschließend den Vertex Shader in `skinning.glvs`, sodass der Vertex mit den Skinning-Matrizen der vier einflussstärksten Knochen transformiert und das Ergebnis mit den entsprechenden Gewichten geblendet wird (Linear Blend Skinning, 1 Punkt).

Abgabe

Verpacken Sie alle Quelldateien des `src` und des `glsl`-Ordners sowie die Ein- und Ausgabedateien von Pinocchio in ein ZIP-Archiv und laden Sie sie in Opal hoch. Achten Sie darauf, dass der Quellcode auf den zur Abnahme verwendeten Rechnern lauffähig ist (Windows, Visual Studio $\geq$ 2015). Zum Testen stehen Ihnen die Rechnerpools der Fakultät zur Verfügung. Sprechen Sie ggfs. mit Ihrem Tutor.