BEUTH HOCHSCHULE
FÜR TECHNIK
BERLIN
University of Applied Sciences

Prof. Dr.-Ing. Kristian Hildebrand

Computergrafik 2 / Aufgabe 2.2
Meshes

WiSe 2015/2016

Bei dieser Aufgabe arbeiten Sie mit dem bereits erzeugten SourceCode der letzten Teilaufgabe. Dabei sollen diesmal aus den Punktwolken zusammenhängende Dreiecksnetze erzeugt werden. Zudem erweitern Sie Ihre Three.js Erfahrungen.

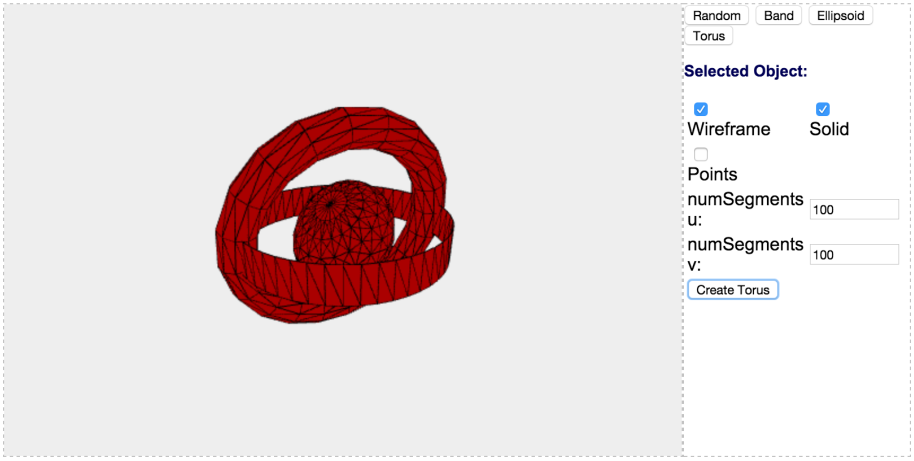

Abbildung 1: Beispielhafte Ergebnisausgabe

Aufgabe 1: Erzeugen der Index-Buffer

Setzen Sie sich mit dem *Three.js*-Modul *BufferGeometry* genauer auseinander. Es ist nicht nur möglich Attribute wie *position* oder *color* hinzuzufügen (über *addAttribute*, wie in der letzten Teilaufgabe benutzt), sondern auch ein IndexArray zu setzen. Das IndexArray muss mit den Vertex Positionen korrespondieren. Drei Indices in dem Array zeigen auf Vertex-Positionen aus dem PositionsArray und ergeben ein Dreieck. (Wie in der Vorlesung besprochen.)

Finden Sie heraus wie das zu implementieren ist und fügen Sie die Funktionalität zu unserem *BufferGeometry*-Modul hinzu.

Fügen Sie in ihren Modellen analog zu den Positionen und Farben ein IndexArray vom Typ *new Uint32Array* mit der richtigen Länge hinzu und füllen das Array mit den richtigen Indices. Fügen Sie desweiteren einen *getter* hinzu, der das IndexArray zurückgibt.

Implementieren Sie im *HtmlController* die Möglichkeit neben den Attributen auch die Indices zu setzen.

Für das Erreichen einer 1.0 ist es notwendig, dass alle Ihre Objekte fehlerfrei als Dreiecksnetz dargestellt werden können.

Aufgabe 2: Arbeiten mit Materialien

Damit die Oberfläche korrekt dargestellt werden kann, muss man *three.js* mitteilen, nicht mehr nur Punkte zu zeichnen, sondern Dreiecke (ein *THREE.Mesh*). Passen Sie *BufferGeometry* dementsprechend an.

Jeder Liste von Primitiven (Mesh, Points) in *three.js* kann man ein Material übergeben. Bislang ist das ein *THREE.PointsMaterial*.

Passen Sie den SourceCode in *BufferGeometry, HtmlController, index.html* so an, dass man dem Objekt bei der Erzeugung auch ein Material zuweisen kann, sodass es als Oberflächen-Model und auch als WireFrame-Model (Linienmodel) dargestellt wird. Setzen Sie sich bitte selbstständig mit der *three.js* Dokumentation auseinander, wie man das machen kann. Wenn Sie darüber hinaus andere Materialien hinzufügen wollen, können Sie das gern machen. Um einem Objekt mehrere Materialien zuzuweisen, können Sie auch *THREE.SceneUtils.createMultiMaterialObject* nutzen.

Zusatzaufgabe (keine Meshes) (nur für das Erreichen einer 1.0) :

Gegeben ein Kreis mit dem Mittelpunkt P_1 und Radius r und Punkt P_2. Wie berechnet man die Punkte P_3 und P_4, die so auf dem Kreis liegen, dass die Tangente durch die Punkte und P_2 geht. Implementieren Sie eine Javascript Funktion und machen Sie diese über einen Button in der *index.html* ausführbar. Nutzen Sie zum Debuggen die JS-Console im Browser oder das Framework aus der 1.Übung. Sie können das auch gern in den alten Übungscode einbauen. Testen Sie Ihren SourceCode mit unterschiedlichen Eingaben.

```
1
2  var tangentPoints = function(radius, p1, p2) {
3
4    var p3 = [0,0];
5    var p4 = [0,0];
6
7    // compute p_3 and p_4 here
8
9    return { p3: p3, p4: p4}
10 };
```

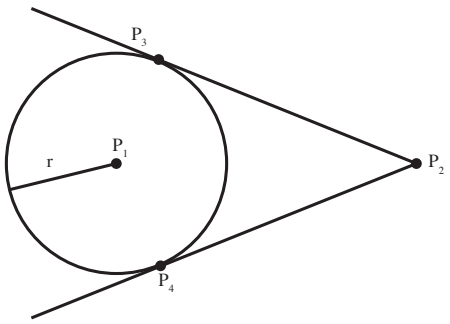

Abbildung 2: Wo liegen die Punkte P_3 und P_4

Abgabe Dies ist *Teilaufgabe 2.2*; die Bearbeitungszeit der Teilaufgabe ist für ca. eine Woche ausgelegt. Die Abgabe der gesamten Aufgabe 2 soll via Moodle bis zu dem dort angegebenen Termin erfolgen. Verspätete Abgaben werden mit einem Abschlag von 2/3-Note je angefangener Woche Verspätung belegt. Geben Sie bitte pro Gruppe jeweils nur eine einzige .zip-Datei mit den Quellen Ihrer Lösung ab.

Demonstrieren und erläutern Sie Ihre Lösung in der nächsten Übung nach dem Abgabetag. Die Qualität Ihrer Demonstration ist, neben dem abgegebenen Code, ausschlaggebend für die Bewertung! Es wird erwartet, dass alle Mitglieder einer Gruppe anwesend sind und Fragen beantworten können.