BEUTH HOCHSCHULE
FÜR TECHNIK
BERLIN
University of Applied Sciences

Prof. Dr.-Ing. Kristian Hildebrand

Computergrafik 2 / Aufgabe 3
Shaderprogrammierung

WiSe 2015/2016

In dieser Übung machen Sie sich mit der Shaderprogrammierung in *Three.js* und *GLSL* vertraut und implementieren zwei visuelle Effekte. Laden Sie den Sourcecode zu Aufgabe 3 von Moodle herunter und kopieren Sie die Dateien so, dass Sie Ihr altes Framework wieder nutzen und ergänzen können. Denken Sie daran, die neuen Module in *app.js* zu registrieren. Die Sourcecode Dateien enthalten einige Anmerkungen und Hinweise zur Implementierung, sind aber ansonsten fast leer. Sie sind also weitestgehend auf sich gestellt und sollten deshalb umso stärker die Übung nutzen, um Problem zu klären.

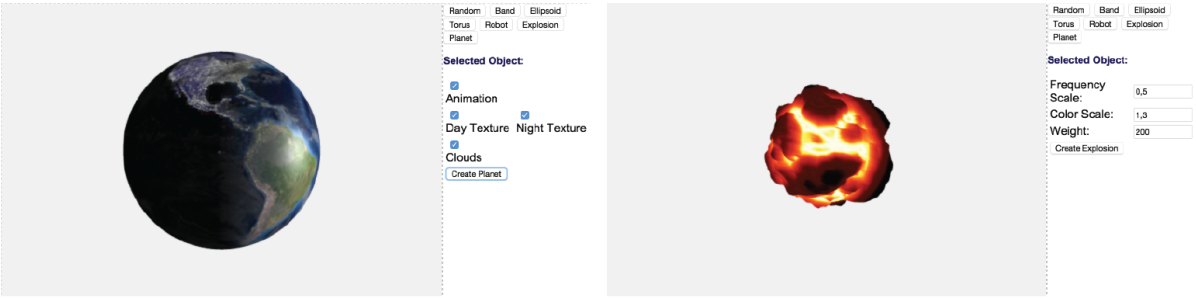

Abbildung 1: Ergebnisausgabe Aufgabe 1 (links). Ergebnisausgabe Aufgabe 2 (rechts).

Aufgabe 1: Beleuchtungsrechnung und Texturierung

In der ersten Aufgabe starten Sie mit einer Kugel. Ziel der Aufgabe ist es, einen Erdglobus unter beliebigen Sonnenständen und mit verschiedenen Oberflächeneigenschaften darstellen können. Dabei können Sie sich einiger NASA-Bilder als Texturen bedienen: Nacht, Tag oder Wolken.

Fügen Sie dazu zunächst unser Planetenmodell und eine ambiente und direktionale Lichtquelle zu der Szene hinzu.

```
1     var planet = new Planet();
2     scene.addMesh(planet.getMesh());
3
4     var aLight = new THREE.AmbientLight( coloe );
5     scene.addLight( aLight );
6     var dlight = new THREE.DirectionalLight( color, intensity );
7     dlight.name = "dLight";
8     dlight.position.set( -1, 0, -0.3 ).normalize();
9     scene.addLight( dlight );
```

Implementieren Sie nun als erstes das Phong-Beleuchtungsmodell, wie es in der Vorlesung vorgestellt wurde, allerdings nun als Funktion im Fragment-Shader. Bereiten Sie also alle Daten im Vertex-Shader auf (Positionen im richtigen Koordinatensystem und Weiterleitung der Texturkoordinaten), sodass sie im Fragment-Shader für die Berechnung genutzt werden können. Nutzen Sie dafür die Anmerkungen in *planet_vs.glsl* und *planet_fs.glsl*.

Ergänzen Sie den Sourcecode auf Applikations/Client-Seite (Anmerkung: der GLSL-Shadercode wird auf der Grafikkarte - dem Server - ausgeführt, die Konfiguration des Shaders wird von der Client-Applikation

durchgeführt). Machen Sie sich mit der Implementierung des *THREE.ShaderMaterial* und mit dem Umgang von Uniform Variablen vertraut und schauen Sie sich folgende Module und Funktionen genauer an: *THREE.UniformsUtils.merge*, *THREE.UniformsLib*. Hinweis: Übergeben Sie die notwendigen Materialparameter (k_a, k_d, k_s, α) als uniforms an den Shader. Nutzen Sie ambientes und direktionales Licht, so wie in der Vorlesung behandelt und fügen den notwendigen Code hinzu.

Wenn Ihre Implementierung richtig ist, sollte ihr Ergebnis ungefähr so aussehen wie in Abbildung 2 dargestellt.

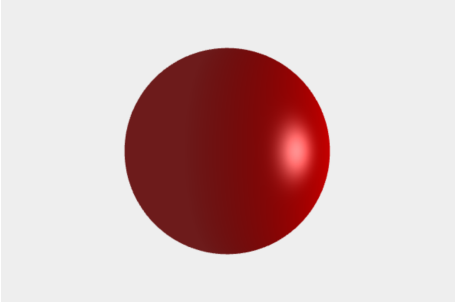

Abbildung 2: Per-Fragment Phong Shader

Laden Sie nun die notwendigen Texturen (mindestens drei) und übergeben diese dem ShaderMaterial als Uniform-Variablen. Binden Sie die GUI-Checkboxes mit ein (im html und auch als Uniform-Variablen), sodass Sie im Shader zur Laufzeit überprüfen, ob eine bestimmte Texture zur Berechnung benutzt werden, oder ob auf die default-Materialparameter (k_a, k_d, k_s, α) zurückgegriffen werden soll.

Hinweis: Überprüfen können Sie das korrekte Laden und Anzeigen einer Textur, indem Sie folgenden FragmentShader nutzen (Liest mittels der Texturkoordinate den rgb-Wert an der Stelle vUv.xy aus und setzt damit die *gl_FragColor*. Bei korrektem Laden, ist die Texture auf der Kugel zu sehen.):

```
varying vec2 vUv;
uniform sampler2D texture;

void main() {
  vec3 color = texture2D(texture, vUv).rgb;
  gl_FragColor = vec4(color, 1.0);
}
```

Bauen Sie nun die Farbwerte aus den Texturen (so wie in den Codeanmerkungen angegeben) mit in das Phong-Beleuchtungsmodell ein. Ziel ist es, dass eine Textur stärker sichtbar wird, wenn Sie durch die Lichtquelle direkt beleuchtet wird. (entweder man sieht die eine, z.B. die Tagestextur oder die anderen beiden Texturen (Nacht und Wolken) stärker)

Implementieren Sie abschließend eine Rotation der Lichtquelle, um den Planeten (entlang des Äquators). Diese Animation soll über die GUI kontrollierbar sein.

Das Ergebnis der Implementierung ist auf Abbildung 1 erkennbar.

Aufgabe 2: Explosionseffekt

Ziel der Aufgabe ist es, den in Abbildung 1 dargestellten animierten Explosionseffekt zu implementieren. Dazu ist Ihnen Skelettcode vorgegeben (explosion.js, explosion_vs.glsl, explosion_fs.glsl).

Fügen Sie den Skelettcode in ihr existierendes Framework ein. Dazu müssen Sie *app.js* und *shaders.js* updaten und eine Explosion GUI-Section im *.html* einfügen.

Modifizieren Sie nun in *explosion.js* das *THREE.ShaderMaterial* und vervollständigen sie die Uniform-Variablen, die an den Shader übergeben werden sollen. Implementieren Sie den Vertex- und Fragmentshader entsprechend der Kommentare im Sourcecode.

Bauen Sie eine Animation über die Implementierung einer veränderlichen Uniform-Zeitvariable ein, die sich zur Laufzeit ändert z.B.:

```
// start wird zur Laufzeit mit Date.now() initialisiert.
explosion.material.uniforms[ 'time' ].value = .00035 * ( Date.now() - start )
```

Zusatzpunkte: Ein einfaches Interface zum Laden von Texturen in *three.js* ist *THREE.ImageUtils.loadTexture(filename)*. Das ist jedoch *deprecated*. Eine korrekte Implementierung wäre hier die Nutzung von Javascript *Promises* - *https://api.jquery.com/category/deferred-object/*. Für eine korrekte Implementierung unter Nutzung von Promises in beiden Aufgaben können Sie Ihre Übungsnote um 0.3 Notenpunkte verbessern. (ja - man kann also auch eine 0.7 als Bestnote in dieser Übung erreichen.)

Hinweis: Nutzen Sie die Übungen für Rückfragen und eventuell weitere Erläuterungen.

Abgabe Dies ist *Übungsaufgabe 3*. Die Bearbeitungszeit der Aufgabe ist für ca. zwei-drei Wochen ausgelegt. Die Abgabe der Aufgabe 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.