

## Curv3D – Submission 2

### Kamera

Die Kamera ist hinter dem Hovercraft angebracht und bewegt sich zusammen mit dem Hovercraft beständig nach vorne. Mit den Tasten A und D kann man den Winkel der Bewegung anpassen.

Durch Klick auf „F1“ wechselt man auf eine frei bewegbare Kamera, welche man mit den Tasten W, A, S, D und der Maus steuern kann.

### Moving Objects

Das Moving Object in unserem Spiel ist das Hovercraft. Die Vorwärtsbewegung von diesem ist strikt vorgegeben und seine Geschwindigkeit erhöht sich im Laufe des Spiels. Mithilfe der Tasten A bzw. D kann dreht sich das Hovercraft nach links bzw. rechts. Mit den Tasten W bzw. S kann man die Höhe des Hovercrafts ändern.

### Model Loading & Texture Mapping

Für den Prototyp wurde Model Loading inklusive Texture Mapping bereits implementiert. Models werden hierbei über die Assimp Bibliothek geladen. Da ausschließlich mit dem Wavefront OBJ Dateiformat gearbeitet wird, wurde auch nur dieses Format getestet. Objekte die zu sehen sind, wurden in Maya inklusive Materialien und UV Mapping gezeichnet. Derartige Daten werden ebenfalls von Assimp über die zugehörige Material (.mtl) Datei eingelesen. Da ein Model aus mehreren Meshes bestehen kann, wurde hierfür die Klasse Mesh definiert. Wie schon erwähnt, werden Informationen zu den Texturen aus der .mtl Datei extrahiert. Dadurch können alle Texturen eines Models über die SOIL Bibliothek geladen werden.

Bevor Model Loading implementiert wurde, wurden primitive Objekte (Würfel) verwendet. Hierfür wurde ursprünglich die TextureLoader Klasse verwendet um Texturen anzubringen.

### Lightning

Als Lichtquelle wird ein Directional light (die Sonne) mit einem ambienten, diffusen und spekularen Anteil verwendet. Das verwendete Belichtungsmodell ist das Blinn-Phong Belichtungsmodell. Für die Objekte wird sowohl für den ambienten, als auch den diffusen Anteil, der diffuse Anteil der Textur verwendet. Für den spekularen Teil wird der Halfway-Vector berechnet. Die Cubes und die Sonne werden nicht beleuchtet.

### Controls

Tasten	Aktion
W, S	Hovercraft nach oben/unten bewegen
A, D	Hovercraft nach links/rechts wenden
E	Perspektivewechsel
Q	Wechsel auf frei bewegbare Kamera
W, A, S, D und Maus	Bewegung der frei bewegbaren Kamera
F1	Help
F2	Frametime (on/off)

F3	Wireframe (on/off)
F6	Bloom (On/Off)
F7	Lens Flares (On/Off)
Esc	Beenden

## Gameplay

Dem Spieler wird die Kontrolle über ein Hovercraft übergeben. Dieses bewegt sich automatisch vorwärts und hinterlässt auf seinem Weg eine Linie aus Cubes. Der Spieler muss geschickt den Hindernissen der Umgebung (Kakteen, Steine) und der selbst gezogenen Linie ausweichen. Sollte das Hovercraft in Berührung mit einem der Elemente kommen, ist das Spiel vorbei und in der Konsole wird „Game Over“ ausgegeben. Mittels der Tasten A und D kann der Spieler das Schiff nach links bzw. rechts steuern. Mit den Tasten W und S hat der Spieler die Möglichkeit seine Höhe begrenzt anzupassen (Man kann nicht so hoch/tief fliegen wie man möchte). Das Spiel ist als eine Art „Trainingsmodus“ zu sehen, in dem ein Spieler seine Fähigkeiten verbessern kann, um in den (vlt) später existenten Multiplayer-Modus einen Vorteil zu haben. Ziel des Spiels ist es so lang als möglich durchzuhalten.

## Effekte

Es wurden folgende Effekte umgesetzt:

### *Shadow Maps with PCF*

Alle Objekte, ausser der Line und der Sonne besitzen Schatten. Für die Shadow Map wird in einem ersten Schritt eine DepthMap aller Objekte (ausser den Cubes und der Sonne) aus der Sicht der zentralen Lichtquelle erstellt und in einen Framebuffer gespeichert. Da wir nur ein directional light in unserer Szene besitzen, verwenden wir eine orthografische Perspektive. Danach werden auf Basis der Shadow Map und der LightSpaceMatrix die Schatten der Objekte berechnet. Dazu wird im Fragment Shader ein Schattenwert berechnet, der entweder 1 beträgt sollte der Punkt im Schatten sein oder 0 falls nicht. Dieser wird dann mit dem diffusen und spekularen Anteil des jeweiligen fragments multipliziert. Der ambiente Teil bleibt unverändert. Mittels eines bias wurde Shadow Acne verhindert. Peter Panning durch Front Face Culling. Als PCF wurden einfach die umliegenden texel zur depthmap gesamplet und der Durchschnitt gebildet.

Source: <https://learnopengl.com/#!Advanced-Lighting/Shadows/Shadow-Mapping>

### *Bloom/Glow*

Die Line und die Sonne erhalten einen Bloom Effekt. Dafür wird unsere gesamte Szene in eine HDR Texture gerendert. Aus dieser Texture werden nur die Objekte mit einer bestimmten Helligkeit genommen und mittels eines Gaussian Blur geblurt. Diese geblurte Textur wird ebenfalls in einen Colorbuffer gespeichert. Am Ende wird die Texture der gesamten Szene mit Texture der geblurten Objekte zusammengeführt/geblendet.

Source: <https://learnopengl.com/#!Advanced-Lighting/Bloom>

### *Lens Flares*

Bei Blick in die Sonne erscheinen simple Lensflares (Mit „Q“ in die Free Moveable Kamera wechseln und Blick auf die Sonne richten). Diese werden anhand der hellen Stellen unserer Szene erstellt. Dafür werden von der Textur der gesamten Szene nur helle Stellen entnommen,

gespiegelt und mit einem Faktor multipliziert. Das Ergebnis wird in einen Colorbuffer gespeichert. Dieser wird danach, ebenso wie bei Bloom mit der Textur der Szene geblendet.

Source: <http://john-chapman-graphics.blogspot.co.at/2013/02/pseudo-lens-flare.html>

### **Particles**

Es werden lebende Partikel in einem Buffer der GPU übergeben um einen Düsenstrahl hinter dem Flieger zu simulieren. Um die Partikel zu zeichnen wurde Instancing umgesetzt. Im Debug Modus können jedes Frame ca. 1500 Partikel gezeichnet werden, während im Release Modus bis zu 100.000 Partikel bei ca. 90 fps möglich sind (Geforce GTX 1070). Um eine möglichst stabile Abgabe einzureichen wurden die Partikel nun auf 50.000 reduziert.

<http://www.opengl-tutorial.org/intermediate-tutorials/billboards-particles/particles-instancing/>

### **Complex Objects**

Als komplexe Objekte haben wir in unserem Spiel das bewegliche Hovercraft, diverse Kakteen und Steine, die Sonne, modifizierte Cubes in der Line und unser Areal mit den großen Steinen am Rand.

### **Experimenting with OpenGL**

Es wurden an mehreren Stellen Framebuffer und Colorbuffer verwendet.

### **Features**

Es besteht die Möglichkeit während des Spiels die Perspektive zu ändern.

### **Additional Libraries**

SOIL, Assimp, Bullet

### **Tools for Models**

Maya