Nina Immenroth, 552932
Markus Daniel Mayer, 568470
Konrad Schön, 568240

Praktikumsbericht Computergrafik

Im Rahmen des Moduls Computergrafik sollten wir ein selbstgewähltes Projekt in Form eines Praktikums bearbeiten. Wir entschieden uns, einen gemütlichen Raum mit einem Tisch, der für ein Pen-&-Paper-Rollenspiel gedeckt ist, zu modellieren. Zudem wollten wir die modellierte Szene über Internet-Browser abrufbar machen, weswegen wir als API die JavaScript-Programmierschnittstelle WebGL wählten. Diese basiert auf OpenGL ES und ermöglicht trotz einiger Einschränkungen relativ schnelle Darstellung von 3D-Grafiken im Browser. Als Framework wählten wir Three.js, da es gute Dokumentation und eine große Beispielbibliothek bietet. Einige Modelle im gltf-Format erhielten wir von Sketchfab.com.

Implementierung

index.html

In diesem html-File wird der Titel der Browseransicht festgelegt, sowie die Bildbreite und -höhe auf 100% und der Rand auf 0 gesetzt.
Anschließend werden nacheinander die JavaScript Dateien aufgerufen, deren Funktionen im Folgenden beschrieben werden.

three.js [1]

Diese Datei enthält die Three.js Bibliothek, welche wir als Framework genutzt haben.

WebGL.js [2]

Diese Datei prüft, ob der verwendete Browser oder die Grafikkarte WebGL unterstützt und gibt eine Fehlermeldung aus, wenn nicht.

OrbitControls.js [3]

Mit den in dieser Datei enthaltenen Funktionen wird ermöglicht, mit der Kameraansicht etwas zu umkreisen (mit linker Maustaste bzw. einem Finger auf dem Touchpad), zu zoomen (mit mittlerer Maustaste bzw. zwei Finger aufeinander zu oder von einander weg bewegen auf dem Touchpad) oder die Kamera zu schwenken (mit rechter Maustaste und ctrl/meta/shift-Taste, oder den Pfeiltasten, oder mit zwei Fingern auf dem Touchpad).

TextureLoader.js [4]

Lädt die Texturen.

GLTFLoader.js [5]

Lädt die GLTF-Dateien.

backgroung.js

Nina Immenroth, 552932
Markus Daniel Mayer, 568470
Konrad Schön, 568240

Mit dieser Datei werden erst die Wände des Raums gebaut und texturiert und anschließend einige Hintergrundmöbel eingefügt. All das ist in der Funktion **create_surroundings()** implementiert. Zuerst wird die Funktion **create_background()** aufgerufen. Hier wird zunächst eine Box erstellt, die unseren Raum darstellen soll. Nun werden nach und nach verschiedene Texturen [6-8] für Boden, Wände und Decke mithilfe eines TextureLoaders geladen und jeweils in eine Variable vom Typ THREE.MeshLambertMaterial gespeichert, wobei das Argument side: THREE.Backside dafür sorgt, dass die Textur später auf der Innenseite unserer Box, also auf den Wänden unseres Raumes angezeigt wird. Texture.repeat.set(x, y) sorgt dafür, dass immer genug Textur für die ganze Wand da ist. Anschließend werden die zuvor erstellten Variablen in ein Array materials gespeichert, das dann bei der Generierung des Raum-Meshes mit THREE.Mesh gemeinsam mit unserer zu Beginn erstellten Box-Geometrie als Argument übergeben wird, und die Reihenfolge angibt, in welcher die Seiten der Box texturiert werden sollen. Schließlich werden die Position des Meshs und die Eigenschaft Schatten aufzufangen festgelegt und das Mesh wird zur Szene hinzugefügt.
Daanch wreden mit **create_teppich()**, **create_bookshelf()** und **create_door()** jeweils eine planare Fläche („geometry“) der entsprechenden Größe erzeugt, die Textur Teppich, Bücherregal bzw. Tür [9-11] geladen. texture.wrapS und texture.wrapT werden auf THREE.MirroredRepeatWrapping gesetzt, wodurch die Texturen zur Vervielfältigung sowohl horizontal als auch vertikal gespiegelt werden. Dann wird wieder ein Mesh aus Textur und der planaren Fläche erzeugt, das Mesh wird rotiert und positioniert, Schatten werden hinzugefügt und das Mesh wird zur Szene addiert.
Das nun folgende **create_poster(size_w, size_h, pos_x, pos_y, pos_z, rot_x, rot_y, rot_z, path)** funktioniert ebenso, abgesehen davon dass hier Größe, Position, Rotation und Pfad zur Textur [12] als Argumente übergeben werden.
Schließlich werden noch eine Couch, Stühle und drei Lampen aus GLTF-Dateien [13] geladen.

lighting.js

In der **illuminate()** Funktion werden Lichtquellen zu unserem Raum hinzugefügt. Zuerst wird AmbientLight (grauweißes ambientes Licht) erstellt, damit die Schatten nicht schwarz sind. Dann wird mit **create_light_ceiling(color, x, y, z)** das Licht der Deckenlampe als Punktlichtquelle erstellt. Die Entfernungen für dichtesten und am weitesten entfernte Schatten sind 1 und 600, da dies unsere stärkste Lichtquelle ist. Die Shadowmap ist groß, damit wenig Aliasing auftritt. Um die Lichtquelle wird eine Zylinder-Form als Glühbirne erstellt, die 50% heller ist als das Licht, damit es so aussieht, als würde sie leuchten. In der Deckenlampe haben wir drei Lichtquellen verwendet, damit die Schatten des Lampenschirms nicht so hart und dunkel ist.
Mit **createLight(color, size)** werden auf ähnliche Art dreimal jeweils eine Lichtquelle hinzugefügt, wobei sie verschieden gefärbt sind und keinen Schatten werfen. Um diese werden Sphären als Glühbirne hinzugefügt. **createLighttube(color, radius, size, rot_x, rot_y, rot_z)** wird eine Punktlichtquelle mit einem länglichen THREE.CylinderBufferGeometry umschlossen, dessen Material nicht transparent ist.

table.js

create_table(x, y, z, scale) erstellt die Geometry für einen Tisch, welcher im Endeffekt ein Billiardtisch ist und auch die selben Maße hat wie einer
Der Tisch besteht aus 10 THREE.BoxGeometry, welche jeweils mit einer Textur [14,15] versehen werden.

Danach werden mit **create_character_sheet** an sich gegenüberliegenden Seiten Blätter mit Charakter-Informationen hinzugefügt. Sie bestehen aus einer PlaneGeometry, die eine Text-Textur erhält.
create_gm_screen erzeugt einen GameMaster-Screen aus 4 THREE.PlaneGeometry
Das Bild aus dem die Front Texturen bestehen ist Eigentum von Paizo Inc. [16]
Das Bild aus dem die Back Texturen bestehen stammt von Hobbygames.ru. [17]
create_playbook erzeugt THREE.PlaneGeometry und versieht sie mit Textur
Das Bild aus dem die Textur besteht ist Eigentum von Paizo Inc. [18]
Anschließend werden eine Reihe Figuren aus GLTF-Dateien[13] geladen und auf dem Tisch platziert. Diese waren teilweise farblos und wurden nur zusätzlich eingefärbt.

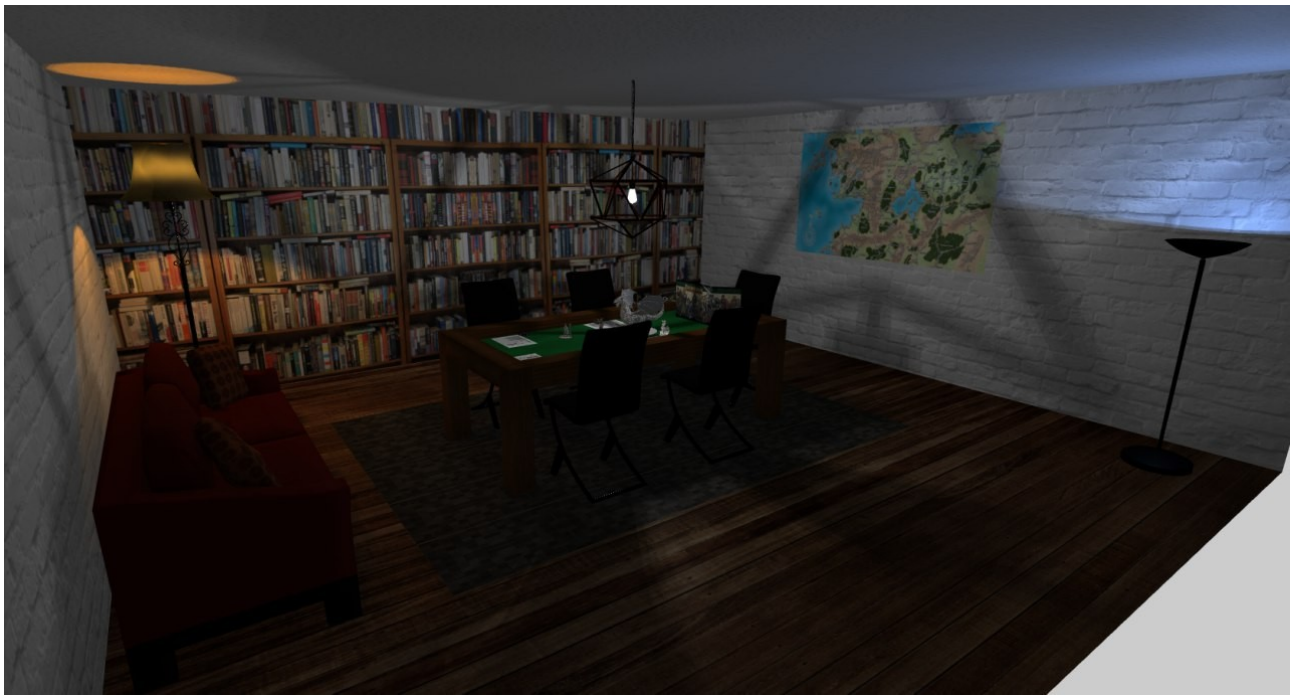
main.js

init() initiiert die Szene mit THREE.Scene() und gibt der Szene eine Hintergrundfarbe. Dann wird ein Renderer mit THREE.WebGLRenderer() erstellt, wobei Antialiasing aktiviert ist. Fenstergröße und Pixelverhältnis werden denen des abspielenden Geräts angepasst. Die Shadowmap wird aktiviert und als Typ THREE.PCFSoftShadowMap festgelegt, da dieser auf Texturen besser aussieht als der Standard Typ.
Die Perspektive der Kamera sowie dass wir 1000 Längeneinheiten weit schauen können wird festgelegt. Wir setzen die Position der Kamera, jedoch nicht die Ausrichtung, da diese standardmäßig auf (0, 0, 0) gesetzt ist.
THREE.OrbitControls wird zum bewegen der Kamera aktiviert.
create_surroundings (aus backgroung.js) und **create_tabletop_table** (aus table.js) sowie **illuminate** (aus lighting.js) werden aufgerufen.
Die **render()** Funktion wird in **animate()** aufgerufen, da immer wieder ein neues Bild berechnet werden muss, wenn sich etwas bewegt.
Schließlich ist noch das Laden von GLTF-Files mit THREE.GLTFLoader() implementiert. Jeder Teil der gltf-Szene wird durchlaufen und geschaut, ob es ein Mesh ist. Wenn ja, werden Schatten hinzugefügt. Dann wird das entstandene Mesh angepasst, positioniert und zu unserem Raum hinzugefügt.

Die main.js ist die letzte Datei im index.html, da hier angefangen wird, Funktionen auszuführen. Dies sollte zuvor vermieden werden, damit erst alle Funktionen da sind, bevor man anfängt, etwas zu berechnen.

Nina Immenroth, 552932
Markus Daniel Mayer, 568470
Konrad Schön, 568240

Ergebnis-Bild



Nina Immenroth, 552932
Markus Daniel Mayer, 568470
Konrad Schön, 568240

Quellen

1) https://threejs.org/
2) https://github.com/mrdoob/three.js/blob/master/examples/js/WebGL.js
3) https://github.com/mrdoob/three.js/blob/master/examples/js/controls/OrbitControls.js
4) https://github.com/mrdoob/three.js/blob/dev/src/loaders/TextureLoader.js
5) https://github.com/mrdoob/three.js/blob/master/examples/js/loaders/GLTFLoader.js
6) https://www.lifeofpix.com/photo/wood-texture-3/
7) https://c1.staticflickr.com/9/8244/8590273977_dab3ccd0bf_b.jpg
8) http://hires.patternpictures.com/PP17272013-Subtle-Plaster-Texture-White-Wall.jpg
9) https://i.pinimg.com/originals/30/a6/b2/30a6b226bf34a6e4dbf405c24fefadaa.jpg
10) https://www.mrjdesigns.co.uk/media/wysiwyg/wall_of_books.jpg
11) http://www.photoshoptextures.com/architectural-textures/white-door-texture.jpg
12) http://3.bp.blogspot.com/-taxbcU3l9eU/VBnCkb_1J5I/AAAAAAAAAO04/BAAw5urvqR8/w1200-h630-p-nu/map_inner_sea_large.png
13) https://sketchfab.com
14) https://www.moebel-und-garten.de/moebel/AS4HOME/Klebefolie-Holzdekor-Moebelfolie-Holz-Eiche-natur-dunkel-45-cm-x-200-cm-Designfolie-von-AS4HOME-104431737.jpg
15) https://us.123rf.com/450wm/yamabikay/yamabikay1705/yamabikay170500039/77519356-gr%C3%BCne-filz-textur-f%C3%BCr-poker-ein-casino-thema-nahtloser-quadratischer-hintergrund-fliese-bereit-hochaufl%C3%B6sendes-fo.jpg?ver=6
16) https://paizo.com/image/content/Secondary/PZO1113-Alt2Secondary.jpg
17) https://hobbygames.cdnvideo.ru/image/cache/hobbygames_beta/data/HobbyWorld/Pathfinder_RPG/Pathfinder_GM_Screen/Pathfinder_GM_Screen_1-1024x1024-wm.jpg
18) https://paizo.com/image/product/catalog/PZO/PZO9036_500.jpeg