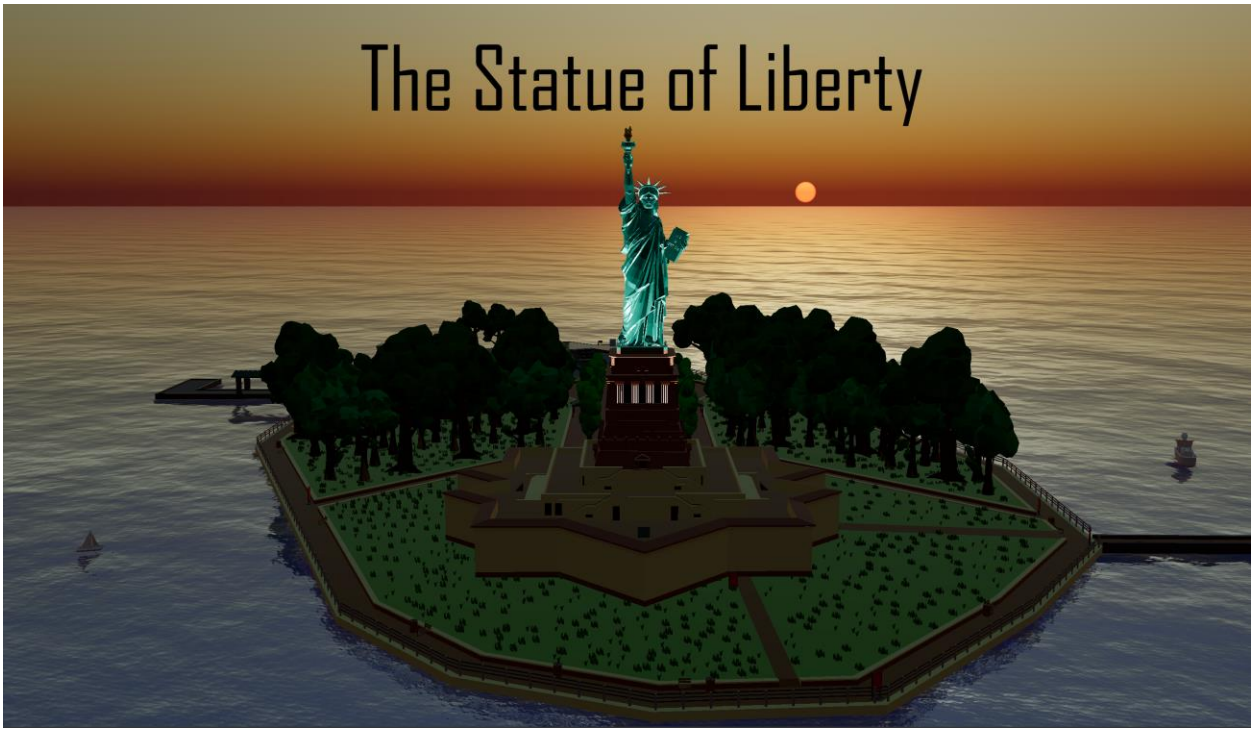
Doku Computergrafik

Liberty Island



Von

Yannic Grafwallner, Christoph Gründer und Gabriel Wuwer

Inhaltsverzeichnis

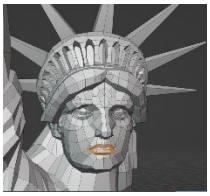
Vorgehen beim Entwurf

Modell Modellierung

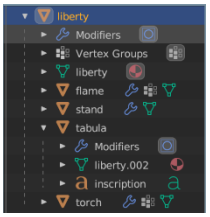
Bei der Modellierung haben wir uns im ersten Schritt für möglichst realistische Modelle entschieden, weswegen diese „Soft shaded“ sind und eine große Anzahl an Polygonen enthalten. Da dies jedoch die Leistung der Anwendung enorm beeinträchtigt, wurde später entschieden auf eine Low-Poly Stilisierung zu setzen und somit eine höhere Objektdichte abbilden zu können.

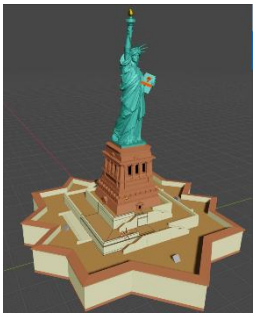
Freiheitsstatue

Die Modellierung der Freiheitsstatue erfolge auf Basis eines vorhandenen Modells (Siehe Hilfsmittelverzeichnis). Dieses war jedoch nicht für die direkte Verwendung geeignet. Folgende Änderungen mussten vorgenommen werden:



- Lose Polygone der Robe der Statue wurden detektiert und Verbunden
- Falsch ausgerichtete Normalenvektoren wurden geflippt
- Das Gesicht wurde zum Großteil neu modelliert, um die gewünschte Ästhetik zu erhalten (Augen, Augenbrauen, Nase, Mund)
- Kollidierende Polygone wurden detektiert und entfernt
- Löcher im Modell wurden gefüllt
- Gesamtmodell wurde in Einzelteile zerteilt, um ein besseres Texturieren und Animation zu gewährleisten (Fackel, Krone, Podest, Tabula, wurden selektiert und in einzelne Objekte zerlegt)
- Texturierung und Materialzuweisung der Statue





Das gesamte große Podest wurde vollständig per Hand modelliert. Dabei wurde zuerst ein Viertel des Sockels modelliert und dieser auf der X und Y-Achse gespiegelt. Ähnlich vorgegangen wurde bei den Treppen. Hierbei wurde eine Stufe modelliert und mit einem „Array Modifier“ in der gewünschten Anzahl wiederholt versetzt dupliziert. Die Treppen wurden danach ebenfalls auf der X und Y-Achse gespiegelt. Die Löcher in den Modellen wurden mit dem „Boolean Modifier“ erstellt.

Umgebung

Die Modellierung der Insel wurde eigenständig per Hand durchgeführt. Als Referenz wurden offizielle Architekturzeichnungen und Luftaufnahmen von Google Maps verwendet. Hierbei wurde die Insel mit hoher Detailtreue an das Original angepasst. Um den zeitlichen Rahmen der Arbeit nicht zu sprengen, floss bei der Gestaltung der umliegenden Gebäude künstlerische Freiheit mit ein und es wurde teilweise vom Original abgewichen. Mehrfahrvorkommende Objekte wurden mit Partikel Effekten modelliert, indem der „Partical Hair“ Effekt genutzt wurde. Dieser Platziert auf eine Fläche ein Objekt oder eine Objektgruppe der Wahl zufällig verteilt mit verschiedenen Größen und Ausrichtungen.

Texturen und Materialien

Die Texturierung wurde überwiegend mit einzelnen Farben oder Materialeinfärbung gelöst, um den LowPoly Stiel gerecht zu werden. Eine Ausnahme bringt das Frachtboot mit sich, welches aufwändige PBR-Texturen enthält. Vor allem das Dach des Bootest beinhaltet eine Albedo, Metalness, Roughness und Normal Map bzw. Textur, welche das Modell realistische wirken lassen. Dies wir vor allem ersichtlich, wenn Licht auf das Dach fällt und verschiedene Bereiche verschieden stark reflektieren und herausstehen, und das, obwohl das Dach eine flache Ebene ist.
Ebenso Kratzer und Abnutzungen auf dem Glas sind über diese Techniken modelliert worden.

Programm Implementieren

Kamera Steuerung

Die Steuerung der Kamera erfolgt über die Three.js-Addons OrbitControls und FlyControls. Die Verwendung beider Klassen ermöglicht das Wechseln zwischen 3 Modis. Im ersten Modus kann die Kamera frei bewegt werden, mit Steuerungen, die einer Drohne sehr nahekommen (pitch, yaw, roll). Im zweiten Modus ist die Kamera auf einer Art Umlaufbahn um die Statue wodurch eine gute Möglichkeit geschaffen wird diese zu betrachten. Im 3. Modus ist die Kamera an einer festen Position, die festen Positionen können dadurch leichter angesteuert werden.

Umgebung

Die Umgebung besteht aus den beiden Hauptkomponenten Skysphere und Water, die dem Three.js Addon-Framework entnommen sind.

Das Einbinden der Nebelfunktion hat sich als nicht sinnvoll erwiesen, da die Skysphere durch diese unbeeinflusst bleibt und dadurch ein harter Übergang zwischen dem Nebeligem Ozean und der davon unbeeinflussten Skysphere entsteht. Dieser Übergang hätte durch das Ermitteln der Farbe der Skysphere am Übergang zum Horizont und darauffolgende Anpassung der Nebelfarbe verhindert werden können, da jedoch auch in diesem Fall Objekte, die über den Horizon hinausragen, bei höherer Distanz die Farbe des Nebels annimmt, wäre das Ergebnis wenig realistisch ausgefallen und wurde deshalb nach eingehender Betrachtung wieder verworfen.

Ein switch in der unteren linken Ecke erlaubt eine voreingestellte Anpassung der Belichtungsparameter und aktiviert die Spotlights um die Statue herum, so dass der Eindruck entsteht es sei Nacht.

Sound design

Tagsüber ist dem Standort entsprechend Möwengeschrei zu hören, während in der Nachtansicht Alicia Keys bezaubernde Stimmung „New York“ (Empire State of Mind (Part II) Broken Down) singt.

Licht

Die Beleuchtung basiert auf monodirektionalem Licht, welches die gesamte Szene aus veränderbarer Richtung erhellt. Die Richtung des Lichtvektors ist hierbei mit der Skybox abgestimmt, so dass die Sonne an der Stelle erscheint, aus der das Licht kommt. Durch Parameter wie raylight und mieDirectionalG kann die Beleuchtung weiter optimiert und so eine optimale Balance zwischen direktem und indirektem Licht, sowie der passende Lichtwärmegrad gefunden werden.

Die Belichtung der Szene kann durch den Parameter „exposure“ nachjustiert werden.

Für die Nacht werden Scheinwerfer aktiviert, die die Statue von unten beleuchten.

Physik

Objektbewegung wird durch Anpassung der Drehung und Verschiebung der bewegten Objekte abhängig von der Zeit erreicht. Die Funktion hierfür wurde selbst entworfen und implementiert.

```
obj.position.z =
startZ+Math.sin(renderZaehler/1000+Math.PI/2+startausrichtung)*radius*20;
```

Die Kollision Detektion ist implementiert mithilfe von Raycasts welche jeden Frame messen wie weit vor der Kamera sich ein Objekt befindet. Sollte ein Objekt vor der Kamera sein und der Abstandsschwellwert unterschritten werden wird dies als Kollision gewertet.

```
function animate() {
    const delta = clock.getDelta();
    if (controls instanceof FlyControls) {
        ray.setFromCamera(new THREE.Vector3(0, 0, 0), camera);
        var collisionResults = ray.intersectObjects(scene.children, true);
        if (collisionResults.length > 0 && collisionResults[0].distance < 10 && Date.now() - timeAfterCollision > 50) {
            collisionDetected("s");
         …
```

Da die Steuerung mithilfe des „FlyControls“ Add Ons implementiert stellte es sich als Herausforderung raus diese zu modifizieren, so dass der Nutzer bei einer Kollision aufgehalten wird in das Objekt reinzufliegen. Hierfür wird die Steuerung deaktiviert und erst wieder aktiviert, wenn der Nutzer vor hat vom Objekt wegzufliegen. Damit nach einer Kollision nicht direkt wieder eine aufgerufen wird weil der Nutzer nicht schnell genug vom Objekt weggeflogen ist, wurde die „timeAfterCollision“ Variable eingeführt. So wird eine Kollision erst 50ms nach der letzten Kollision wieder erkannt. Diese Konzept ist ein Workaround und kann bei gezieltem missbrauch zu Fehlern führen.

```
function collisionDetected(freeCollisionKey) {
    controls.enabled = false;
    controls.dispose();
    console.log("Press " + freeCollisionKey + " to free the camera");
    document.getElementById("status").textContent="Tap S";

    // Wait for keypress s before enabling controls again
    document.addEventListener("keydown", function freeCollision(event) {
        if (event.key === freeCollisionKey) {
            timeAfterCollision = Date.now();
            controls = initControls(camera, renderer, controls);
            controls.enabled = true;
            document.removeEventListener("keydown", freeCollision);
            document.getElementById("status").textContent="Flight";
        }
    });

}
```

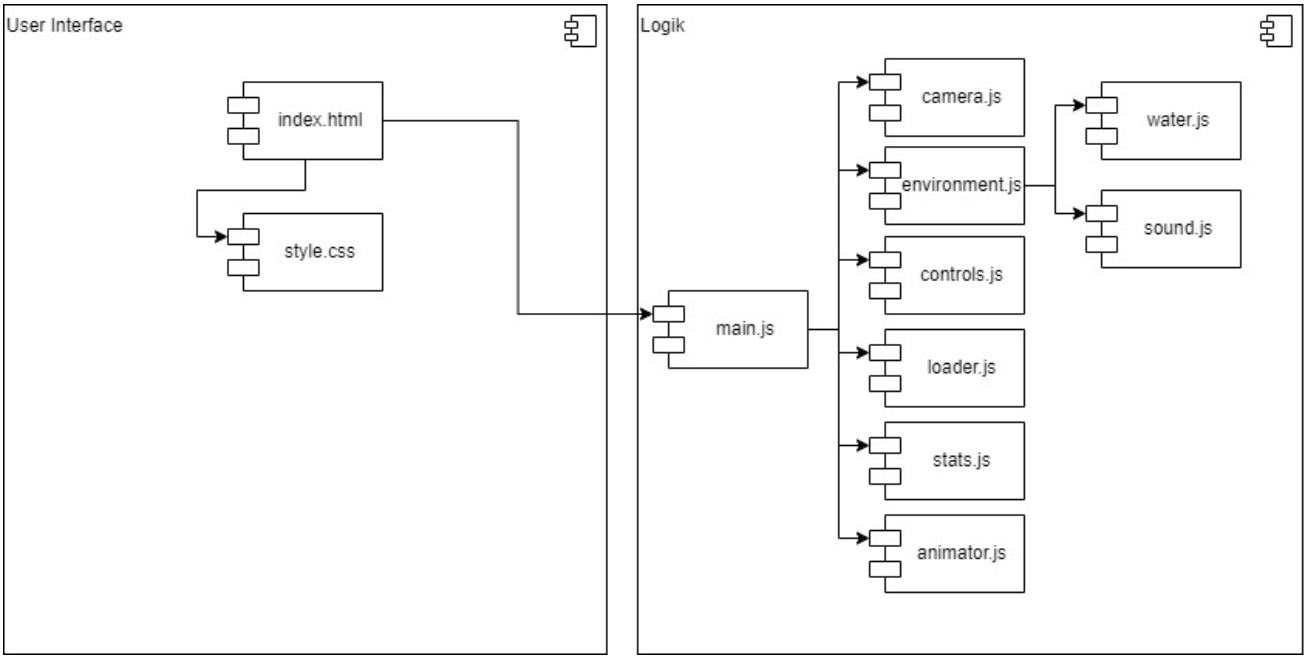
Programmdesign

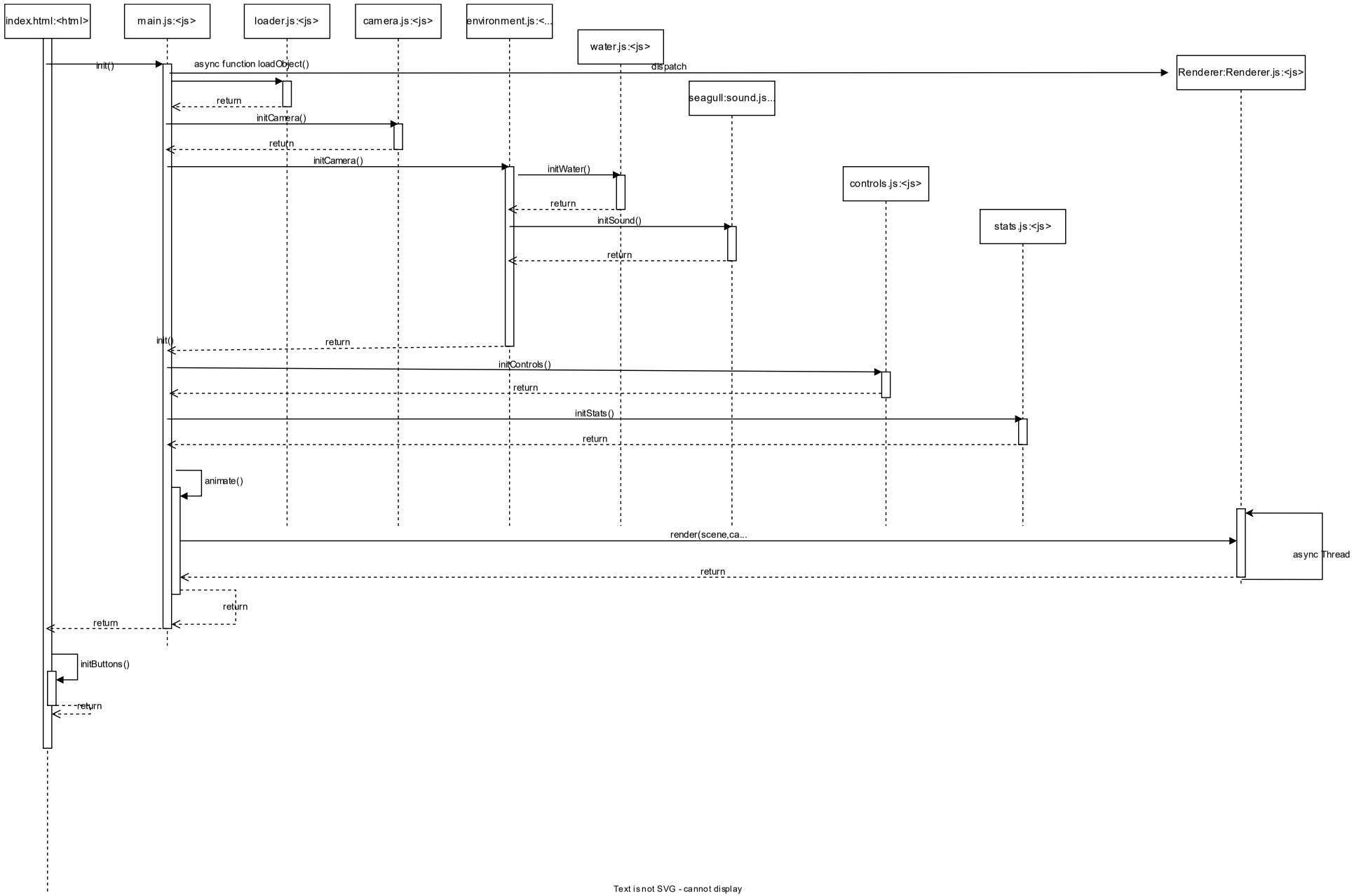
Auslagerung von zusammenhängendem Code in eigene Klassen

Um von der unübersichtlichen „One File for everything“- Praktik wegzukommen, die sich gerade zu Beginn des Projektes eingeschlichen hat, wurde zunehmend inhaltlich zusammenhängender Code in extra Files ausgelagert und durch die „export function“ Deklaration aus dem Main File aufgerufen.

Hierbei war stets auf Anpassung des scopes durch Übergabe in Funktionen bzw. Export der daten aus main zu beachten, um weiterhin volle funktionsweise und späteren zugriff auf die Daten auch nach Ablauf der ausgelagerten Funktion zu gewährleisten.

Diagramme





Text is not SVG - cannot display

Hilfsmittelverzeichnis

Modelle:	
Blender - 3D Modellierung Suite	https://www.blender.org/
Referenzen – Liberty Island	https://archello.com/project/the-statue-of-liberty-museum
Referenzen – Gesamt	https://www.google.com/maps/place/Freiheitsstatue/@40.6896684,-74.0465145,504m/data=!3m1!1e3!4m6!3m5!1s0x89c25090129c363d:0x40c6a5770d25022b!8m2!3d40.6892494!4d-74.0445004!16zL20vMDcycDg?entry=ttu
Referenzen – Freiheitsstatue	https://www.cadnav.com/3d-models/model-40879.html
ThreeJS-Addons:	
Wasser	https://github.com/mrdoob/three.js/blob/master/examples/webgl_shaders_ocean.html
Sky	https://github.com/mrdoob/three.js/blob/master/examples/webgl_shaders_sky.html
Stats	https://github.com/mrdoob/stats.js
FlyControls	https://github.com/mrdoob/three.js/blob/master/examples/misc_controls_fly.html
OrbitControls	https://github.com/mrdoob/three.js/blob/master/examples/misc_controls_orbit.htmlt
GLTF-Loader	https://github.com/mrdoob/three.js/blob/master/examples/webgl_loader_gltf.html

Installationsanweisungen

Siehe README.md im Projektverzeichnis des Projektes.