Projektbericht

T3B732 Multimediatechnologien

Thema: WebGL

Hochschule Karlsruhe – Technik und Wirtschaft

Christoph Hölzer

Matrikelnummer: 33429

Inhaltsverzeichnis

Das Multimediaprojekt

Einleitung

Im Verlauf dieser Projektarbeit (T3B732 Multimediatechnologien) bekam jeder Student die Aufgabe, sich eigenständig mit einer neueren Multimediatechnologie zu beschäftigen und den Stand (der Entwicklung) in diesem Bericht festzuhalten. Zusätzlich galt es festzustellen, welche Rolle diese Technologie in der Technischen Dokumentation spielen kann und welche neuen Wege sich damit in Zukunft beschreiten lassen.

Es standen unterschiedliche Themen zur Auswahl bereit. So konnten wir z.B. Technologien wie Augmented Reality erkunden oder mit einer Virtual Reality Powerwall arbeiten. Zentrales Thema war die Erstellung von Apps und damit das „Experimentieren“ mit verschiedenen App-Buildern. Die meisten Themen waren verknüpft mit den Möglichkeiten einer Umsetzung in HTML5. Es bestand jedoch auch die Möglichkeit, eigene Ideen zu verfolgen.

Da mich 3D und Animation schon immer faszinieren, fiel meine Entscheidung auf das Thema WebGL. Surft man heutzutage im Internet, findet man kaum mehr Seiten ohne Videos. Besonders 3D-Animationen werden immer alltäglicher. WebGL bietet die Möglichkeit, 3D im Browser darzustellen, und zwar ohne dass zusätzliche Plug-Ins nötig sind. In diesem Projektbericht möchte ich darlegen, wie diese recht neue Technologie (erste freigegebene Version März 2011) funktioniert, welche Möglichkeiten sich damit realisieren lassen und welche anderen Werkzeuge im Zusammenspiel mit WebGL eingesetzt werden können.

Projektmanagement

Die Arbeit am Projekt (8 Wochen) war mit einem Umfang von ca. 48 Stunden festgelegt und in der letzten Woche sollte jeder Student in einer kurzen Präsentation sein Thema vorstellen.

Projektplan

Tätigkeit	Von ca.	Bis ca.
Themenwahl	09.10.	16.10.
Einarbeitung in Blender	16.10.	22.10.
Einarbeitung in WebGL	22.10.	27.10.
Einarbeitung in JS	22.10.	25.10.
Testen der WebGL-Frameworks	28.10.	05.11.
3D-Erstellung und Testen der Exportformate	03.11.	15.11.
Erstellen des Projektberichts	19.11.	07.12.

Zeitmanagement

Tätigkeit	Benötigte Zeit (in Stunden)
Themenwahl	3
Einarbeitung in Blender	8
Einarbeitung in WebGL	10
Einarbeitung in JS	3
Testen der WebGL-Frameworks	5
3D-Erstellung und Testen der Exportformate	16
Erstellen des Projektberichts	15
Summe	**60 Stunden**

WebGL und HTML5

Was ist WebGL?

WebGL ist die Abkürzung für Web Graphics Library, also gewissermaßen eine Bibliothek für Web-Grafiken. Als Bestandteil des Webbrowsers ermöglicht es eine native Darstellung (ohne zusätzliche Pulg-Ins) hardwarebeschleunigter 3D-Grafiken. In den Anfangszeiten des Internet undenkbar, ist dies heute mit entsprechend leistungsstarker Grafikhardware möglich.

WebGL basiert auf OpenGL ES (eine „offene Grafikbibliothek“), eine Spezifikation für eine plattform- und sprachenunabhängige Programmierschnittstelle zur Entwicklung von 3D-Computergrafik. WebGL ist also eine vollständig in den Browser integrierte Schnittstelle die den Vorteil mit sich bringt, dass eine Manipulation mittels JavaScript problemlos möglich ist.

Als lizenzfreier Standard von der Khronos Group entwickelt, wurde eine erste Spezifikation von WebGL im Jahr 2011 freigegeben und von den Browsern Chrome, Firefox, Safari und Opera unterstützt. Inzwischen wird WebGL von allen gängigen Browsern unterstützt.

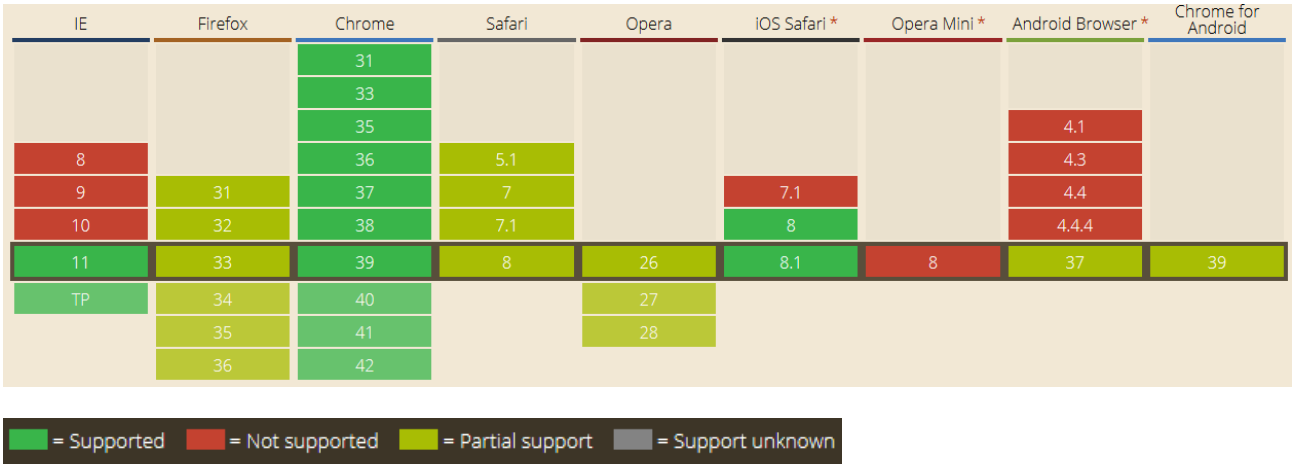

Abb. 1: Browserunterstützung von WebGL

Grundlegende Funktionsweise

Das Canvas-Element

Da WebGL selbst eigentlich kein Bestandteil von HTML 5 ist, erfolgt eine Einbindung über das HTML-Element <canvas>. Die 3D-Inhalte, die im Browser angezeigt werden sollen, werden durch eine JavaScript-Funktion „in das Canvas-Element gerendert“.

Der beschriebene Code ist hier nur auszugsweise dargestellt.

```
<body onload="start()">

  <canvas id="my_canvas" width="640" height="480"> </canvas>

</body>
```

Abb. 2: Canvas-Element im HTML Body

Wenn der Browser das Dokument lädt, wird die JavaScript-Funktion "start" aufgerufen. Im Body befindet sich das Canvas-Element, welches eine Zeichenfläche festgelegt, die als Grundlage für die zu erstellenden WebGL-Inhalte dient. Auf dieser pixelbasierten Fläche kann dann mit JavaScript gezeichnet werden, bzw. es können 3D-Szenen geladen werden.

Bearbeitung mit JavaScript

In der JavaScript-Funktion "start" wird zunächst eine Variable festgelegt, um den Bezug zum Canvas-Element herzustellen:

```
canvas = document.getElementById("my_canvas");
```

Ist der Bezug erfolgreich hergestellt, wird versucht, den sogenannten WebGL-Kontext zu initialisieren:

```
gl = canvas.getContext("experimental-webgl");
```

Geschieht dies ebenfalls erfolgreich (andernfalls muss die Funktion eine Warnmeldung ausgeben), kann auf die WebGL-Schnittstelle zugegriffen werden. Der Kontext wird aktuell noch als „experimental-webgl“ bezeichnet, da die Spezifikation noch in Entwicklung ist.

Damit steht nun ein leeres (rechteckiges) Feld zur Verfügung, das befüllt werden kann. Bevor dies jedoch geschehen kann, werden sogenannte Shader benötigt.

Shader

Shader sind Programme, die vom Grafikprozessor ausgeführt werden. Für WebGL-Inhalte sind zwei Shader notwendig. Um eine 3D-Szene (WebGL) in ein 2D-Bild (Monitor) zu überführen, muss für jeden Pixel ein eigener Farbwert berechnet werden. Dafür ist der Fragment-Shader (oder Pixel-Shader) zuständig. Ein zweiter, der Vertex-Shader, ist für die Manipulation der Geometrie zuständig. Er transformiert die Koordinaten der Oberflächen (eines 3D-Objektes) und definiert so die Position und Form von jedem Punkt. So kann letztendlich die Beleuchtung beeinflusst werden (z.B. Simulation von Wasserwellen).

```
<script id="shader-fs" type="x-shader/x-fragment">

<script id="shader-vs" type="x-shader/x-vertex">
```

Abb. 3: Fragment- und Vertex-Shader

Im Normalfall werden die Shader einem Shaderprogramm angehängt und mit dem Programmcode verlinkt (und kompiliert).

Erstellen und Rendern eines Objekts

Nun kommt das eigentliche Zeichnen. Ein einfaches Beispiel wäre ein Quadrat. Zunächst muss ein Puffer erstellt werden, der die Punkte speichert. Dies geschieht über einen Array, der die Koordinaten aller Punkte enthält und an den Grafikspeicher überträgt (die Berechnungen werden in den Grafikprozessor ausgelagert). Der Puffer wird letztendlich mit dem Kontext verbunden.

```
var vertices = [
  1.0,  1.0,  0.0,
  -1.0, 1.0,  0.0,
  1.0,  -1.0, 0.0,
  -1.0, -1.0, 0.0
];
```

```
var colors = [
  1.0,  1.0,  1.0,  1.0,    // weiß
  1.0,  0.0,  0.0,  1.0,    // rot
  0.0,  1.0,  0.0,  1.0,    // grün
  0.0,  0.0,  1.0,  1.0     // blau
];
```

Abb. 4: Koordinaten (vertices) und Farben (colors) des Quadrats

Mit den fertigen Shadern und dem einfachen Quadrat kann schließlich gerendert werden. Eine weitere JavaScript-Funktion ist verantwortlich für das Zeichnen der Szene und legt z.B. Hintergrundfarbe oder Kameraperspektive fest.

An dieser Stelle müssen z.B. Matrix-Operationen verwendet werden, um komplexere Szenen zu erstellen. Eine große Hilfe stellen hier JavaScript-Bibliotheken (siehe Kapitel WebGL-Frameworks) dar, die eingebunden werden können, um diese Aufgabe zu übernehmen.

Durch eine nachträgliche Änderung der Shader und der Funktion, die die Szene zeichnet, können z.B. neue Farben zugewiesen, Texturen (Laden einer Bilddatei) oder Bewegungen/Animationen hinzugefügt werden. Um z.B. das Quadrat in einen Würfel umzuwandeln, müssen 5 weitere Flächen (über "var vertices") definiert werden. Außerdem können Schatten und Beleuchtung über die Shader hinzugefügt werden.

3D-Erstellung und der Weg in den Browser

Um 3D-Inhalte zu erstellen, gibt es grundlegend zwei Varianten. Man kann einerseits 3D-Grafiksoftware verwenden, um 3D-Objekte selbst zu erstellen oder man nutzt Frameworks, die in eine Website eingebunden werden können.

WebGL-Frameworks

Will man wie im vorigen Kapitel beschrieben eine komplexe 3D-Szene auf einer Website mittels WebGL erstellen, wäre es ein enormer Aufwand, den nötigen Code vollständig selbst zu schreiben. Selbst für ein einfaches Quadrat, das mit ein wenig Bewegung ausgestattet wird, sind hier schon enorm viele Zeilen zu schreiben.

Dieser Aufwand kann mit dem Einsatz von Frameworks stark verringert werden. Im Netz stehen zahlreiche WebGL-Frameworks frei zur Verfügung. Die Khronos Group stellt in ihrem Wiki (https://www.khronos.org/webgl/wiki/User_Contributions) die bekanntesten mit einer kurzen Beschreibung vor.

Hier möchte ich auf die zwei Beispiele Three.js und X3DOM näher eingehen.

Three.js

Three.js ist ein Projekt, an dem zahlreiche Personen beteiligt sind. Es handelt sich um eine 3D-JavaScript-Bibliothek, die von dem webbasierten Hosting-Dienst GitHub kostenlos heruntergeladen werden kann.

In der HTML-Datei wird mit der Zeile <script src="three.min.js"></script> eine Verbindung zu der Bibliothek hergestellt (sofern sie sich im gleichen Verzeichnis wie die HTML-Datei befindet). Funktionen für Szenen, Kameras, Renderer, Animationen, Shader usw. sind in dieser Bibliothek bereits enthalten und müssen deshalb nicht extra selbst erstellt werden.

X3DOM

X3DOM ist ebenfalls ein open-source JavaScript-Framework. Entwickelt vom Fraunhofer-Institut für Graphische Datenverarbeitung, erlaubt es, das Format X3D (Extensible 3D) in HTML einzubinden. Eine Unterstützung durch die gängigen Browser (und entsprechende Plug-Ins) ist gegeben. Die XML-basierte Beschreibungssprache für 3D-Modelle ist ein ISO-Standard und ermöglicht relativ einfache Interaktionen durch den Benutzer.

Im einfachsten Fall können X3D-Dateien mit einem Texteditor erstellt werden. Um sie anzuzeigen, kann auf der Website www.instantreality.org des Fraunhofer Instituts der Instant Player heruntergeladen werden.

3D-Erstellung mit Blender

Die 3D-Grafiksoftware Blender (Download auf www.blender.org) bietet trotz geringer Dateigröße einen relativ großen Funktionsumfang und ist eine kostenlose Alternative zu bekannten, proprietären Programmen wie Maya oder 3ds Max. Basierend auf der Skriptsprache Python kann man (3D-)Körper modellieren, texturieren, animieren und rendern. Die Software ist nur in englischer Sprache verfügbar.

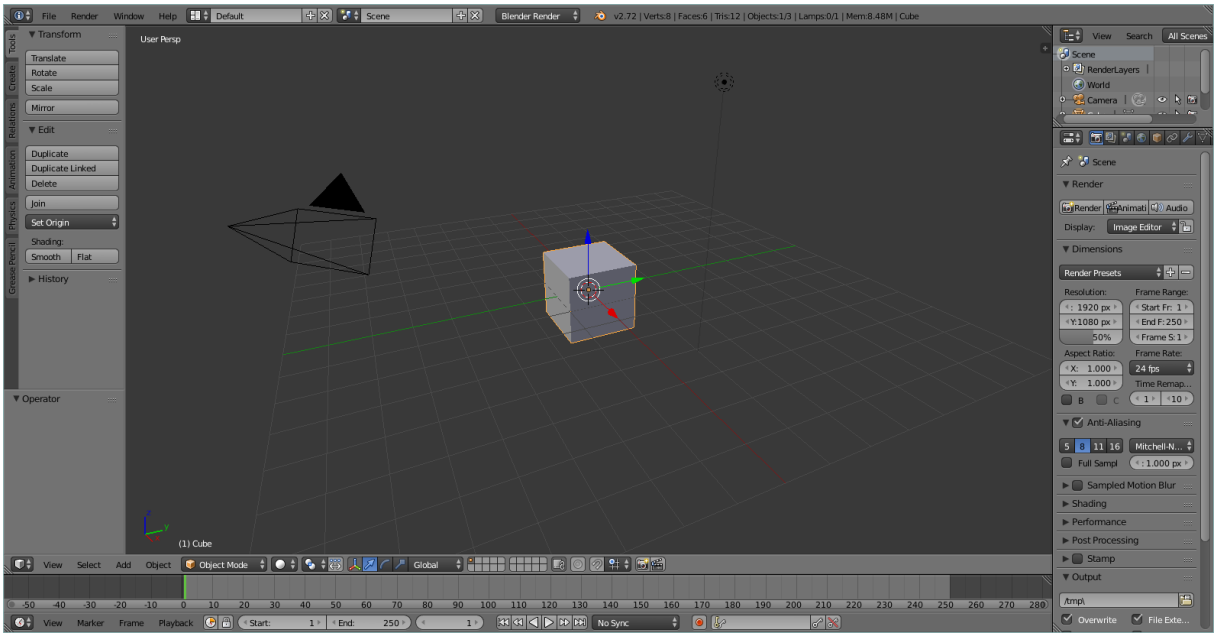

Abb. 5: Die Benutzeroberfläche von Blender

Obwohl Blender ein eigenes Dateiformat „.blend“ verwendet, besteht die Möglichkeit, unterschiedliche Formate zu importieren/exportieren. Die geläufigen Exportformate sind z.B. 3D Studio (.3ds), Wavefront (.obj) und Extensible 3D (.x3d).

Außerdem bietet Blender eine zusätzliche, enorm praktische Funktionalität: Es können verschiedene Add-Ons in Form von Python Scripts hinzugefügt werden, um beispielsweise die Palette der Import- und Exportformate zu erweitern. So kann auch ein Script für einen direkten JavaScript bzw. JSON Import/Export installiert werden.

Im Folgenden werden zwei Varianten beschrieben, für die kein (X3D) bzw. nur bedingter (Three.js) Programmieraufwand nötig ist.

Export mit X3D und Konvertierung in HTML

Eine Methode, um ein mit Blender erstelltes 3D-Objekt zu konvertieren, sodass eine Anzeige im Browser möglich ist, stellt der Weg über den Online-Konvertierer von Instant Reality dar. Dabei wird der X3D- bzw. XML-Code in einfaches HTML umgewandelt.

1. Schritt

In Blender wählt man zunächst als Exportformat "X3D Extensible 3D" und erhält als Ausgabe eine XML-basierte Datei mit der Endung .x3d. Sie kann mit einem einfachen Texteditor angezeigt werden. Die Struktur sieht folgendermaßen aus:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.0//EN"
 <X3D version="3.0" profile="Immersive" xmlns:xsd="
  <head>
        <meta name="filename" content="test.x3d" />
        <meta name="generator" content="Blender 2.72 (sub 0)" />
  </head>
    <Scene>
     ⋮
    </Scene>
 </X3D>
```

Es werden standardmäßig Version, Sprache und DTD festgelegt. Die eigentlichen Inhalte sind (neben dem Head mit Metadaten) innerhalb des Wurzelknotens <X3D> eingeschlossen. Darunter liegt der Knoten <Scene>, in dem entsprechend die 3D-Szene aufgebaut wird.

Die wichtigsten Tags, um eine Szene aufzubauen:
<Transform> bestimmt Position, Ausrichtung und Skalierung der Kindknoten.
<Group> um verwandte Knoten zusammenzuschließen. Ermöglicht eine einfache Wiederverwendung.
<Inline> zur Einbindung weiterer X3D-Szenen.
<Shape> zum Anlegen von 3D-Objekten, mit denen die Szene aufgebaut wird. Enthält einen obligatorischen Geometrie-Knoten (z.B. <Sphere> für eine Kugel) und einen optionalen Knoten <Appearance>, welcher einen weiteren Knoten <Material> für die Festlegung von Farben enthält.

Durch die intuitive Gestaltung der Tags ist eine Änderung der Inhalte relativ problemlos möglich.

2. Schritt

Im zweiten Schritt ruft man die Website
www.doc.instantreality.org/tools/x3d_encoding_converter/ auf.
Unter „1 | choose input encoding“ wählt man das Input Format: XML encoding (X3D).
Nun kopiert man den gesamten Inhalt der aus Blender exportierten X3D-Datei und fügt ihn in das leere Textfeld unter „2 | paste input code“.
Anschließend wählt man unter „3 | choose output encoding“ das Ausgabeformat: HTML5 encoded webpage (x3dom html5).

Klickt man jetzt auf den Button „Convert encoding“, wird die Konvertierung durchgeführt und die Ausgabe in Form von HTML wird im unteren Bereich angezeigt (inklusive Zeilenangaben).
Der erhaltene HTML-Code kann jetzt wieder kopiert und in ein leeres Textdokument eingefügt werden. Zuletzt muss noch mit der Endung .html abgespeichert werden und damit liegt ein funktionsfähiges HTML-Dokument vor.

Alle Inhalte der 3D-Szene sind nun im Body der HTML-Datei eingeschlossen und auch hier kann wieder auf die einzelnen Elemente zugegriffen werden, um Änderungen durchzuführen.

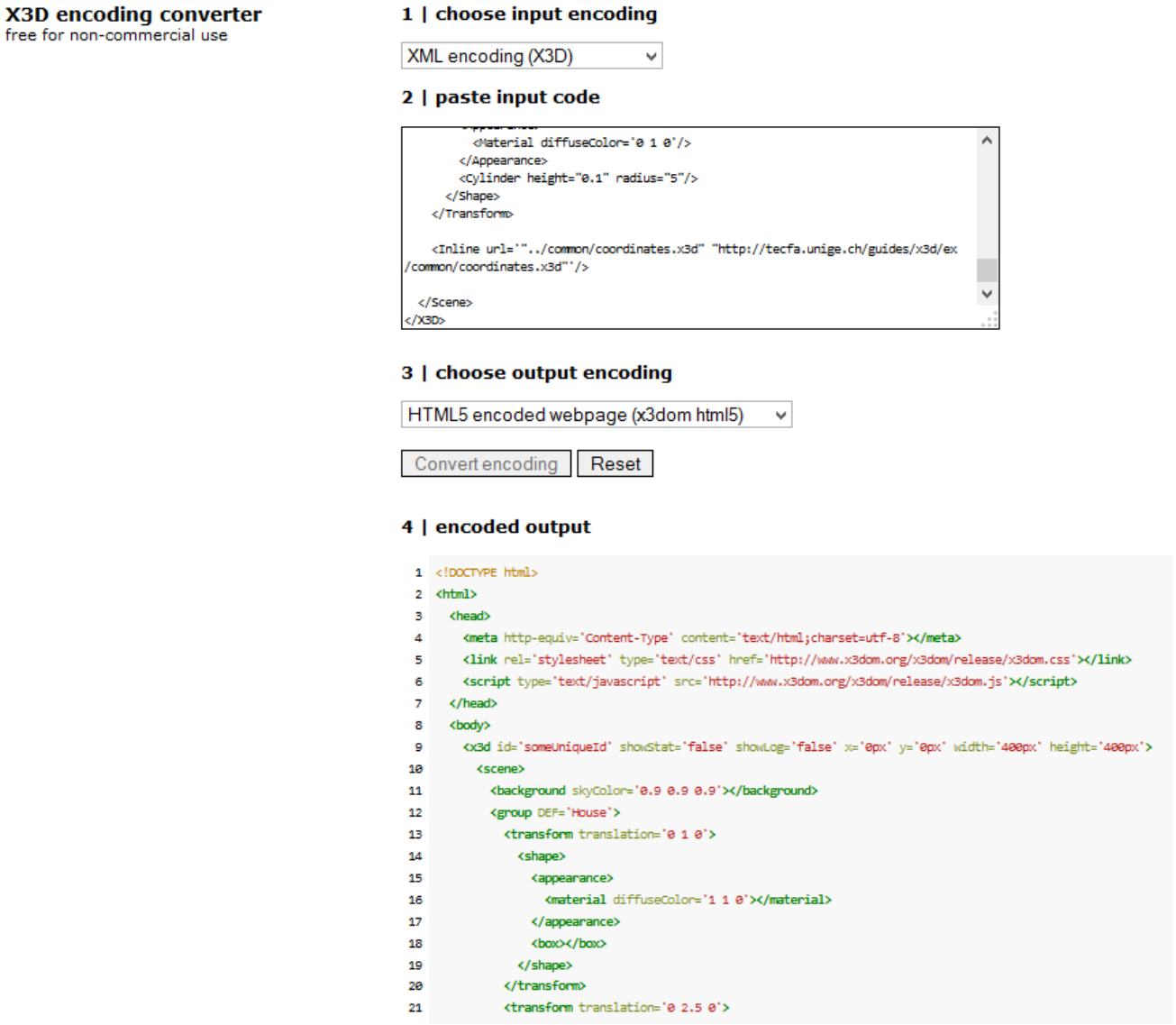

Abb. 6: Der Instantreality Online-Konverter

Export mit Three.js

Im Standardumfang von Blender ist der Export mit dem Framework Three.js nicht möglich, jedoch kann das Programm um diese (und zahlreiche weitere) Funktionalität erweitert werden.

1. Schritt

GitHub bietet die Möglichkeit ein Skript für den Three.js-Import/Export herunterzuladen (www.github.com/mrdoob/three.js/tree/master/utils/exporters/blender). Es kann auch das komplette Three.js-Paket heruntergeladen werden. Da es sich dabei aber um über 100mb handelt und man für die hier beschriebenen Zwecke nur ein bzw. zwei Skripte benötigt, ist dies nicht unbedingt nötig.

Das heruntergeladene Skript kopiert man in das Verzeichnis C:\Users\<username>\AppData\Roaming\Blender Foundation\Blender\<blender_version>\.

2. Schritt

Jetzt öffnet man Blender und wählt im Menü „File“ → „User Preferences…“.

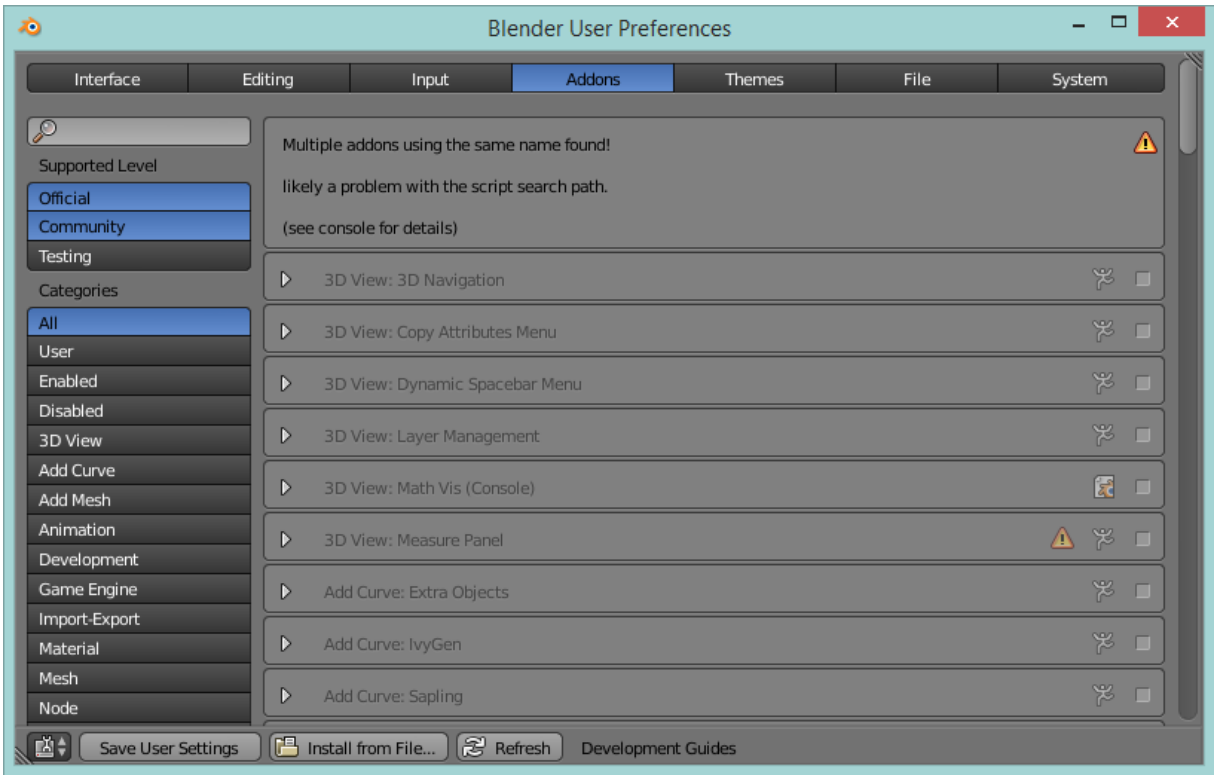

Abb. 7: Blender Benutzerdefinierte Einstellungen

Unter dem Reiter Addons sind alle derzeitig installierten und aktivierbaren Erweiterungen aufgelistet. Auf der linken Seite ist eine Übersicht der verschiedenen Kategorien zu sehen. Dort wählt man „All“ oder „Import-Export“. Wurde das Three.js-Import/Export-Skript im richtigen Ordner abgelegt, erscheint die entsprechende Erweiterung in der Liste.

Als nächstes muss nur noch ein Haken im Feld daneben gesetzt werden und die Funktion ist aktiviert.

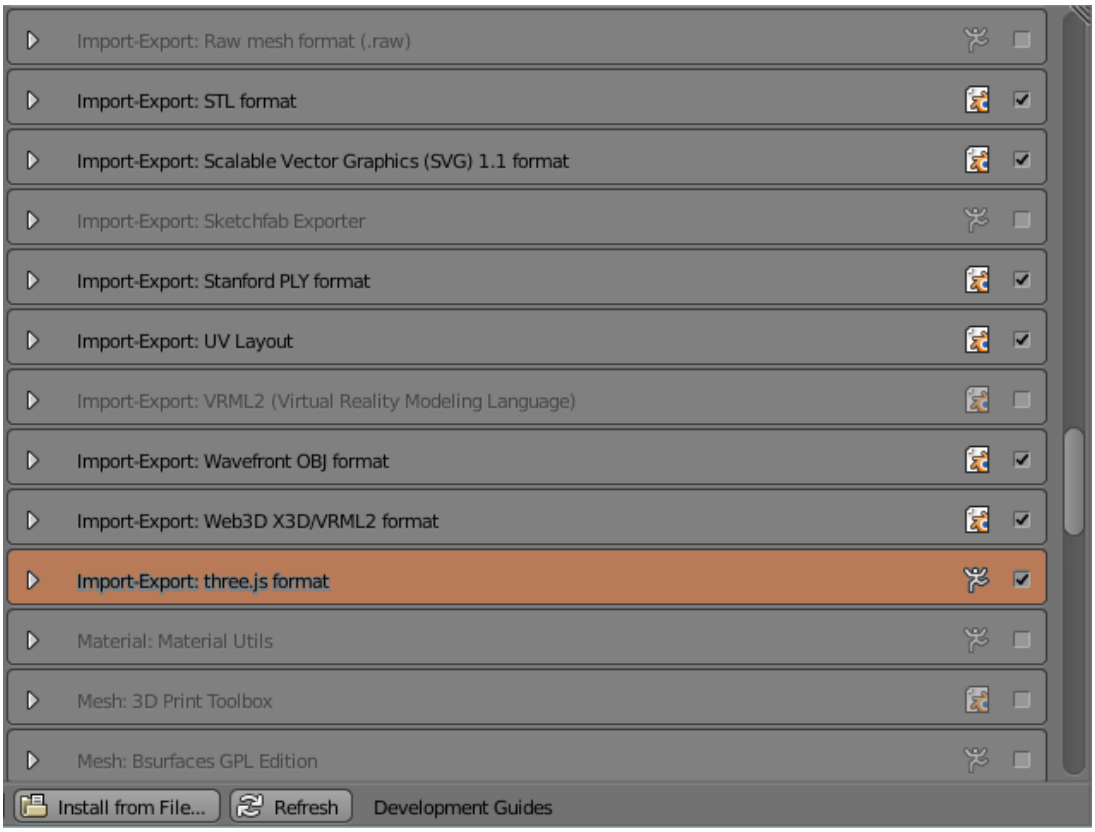

Abb. 8: Das Import-Export-Skript für Three.js wurde hinzugefügt

Hat man diese Schritte erfolgreich durchgeführt, wurde das Export-Menü um das Format Three.js (.json) erweitert.

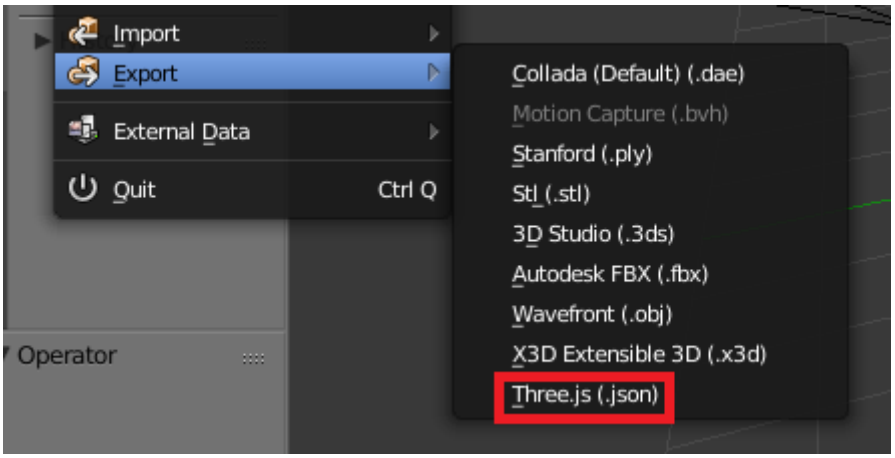

Abb. 9: Die neue Export-Option mit Three.js

3. Schritt

Exportiert man jetzt ein 3D-Objekt mit der neuen Funktion, erhält man eine Datei im Format JSON (JavaScript Object Notation). JSON ist ein kompaktes Datenformat in einer einfach lesbaren Textform zum Zweck des Datenaustauschs zwischen Anwendungen. Wie der Name des Formats schon sagt, wird JSON verwendet, um Informationen wie z.B. Objekte (in diesem Fall das exportierte 3D-Objekt) oder Variablen in einer lesbaren Form abzuspeichern. Dies wird durch die einfache Syntax gewährleistet.

Die exportierte Datei kann z.B. als Variable (var json = {...}) in eine JavaScript-Funktion eingebunden werden.

Fazit

WebGL steckt zwar noch in den Kinderschuhen, dennoch ist unschwer zu erkennen, welche Wege es eröffnet. Die Fähigkeit der hardwarebeschleunigten Darstellung ohne Plug-Ins bringt einen gewaltigen Fortschritt mit sich, betrachtet man die Leistungsfähigkeit, die Browser durch diese Technologie erhalten. Wo vor einigen Jahren PCs noch Probleme hatten manche 3D-Inhalte performant zu verarbeiten, wird nun bereits der nächste Schritt gemacht und es ist möglich, sich im Browser durch virtuelle 3D-Welten zu bewegen. Der Clou besteht darin, dass das Ganze funktioniert, ohne dass etwas installiert werden muss. Des Weiteren ist WebGL natürlich plattformübergreifend, also unabhängig vom Betriebssystem. Selbst Microsoft, welches sich zunächst etwas quergestellt hat, hat sich inzwischen mit WebGL angefreundet.

Betrachtet man die Möglichkeiten des Formats X3D, fällt auf, dass es zu Schwierigkeiten bei der Darstellung fotorealistischer Szenen kommt, was mit den hohen Rechenkapazitäten zusammenhängt. Jedoch zeigt sich hingegen, dass eine Visualisierung von Modellen z.B. im wissenschaftlichen Bereich oder im technischen Bereich Sinn macht. So können mit Blick auf die Technische Dokumentation z.B. Maschinen oder Maschinenteile mit einfachen und schnellen Mitteln dargestellt werden. Dabei kann das Modell für optimale Einblicke beliebig skaliert oder gedreht werden. Geht man einen Schritt weiter können sogar einzelne Elemente hervorgehoben oder virtuell herausgenommen werden. Hier könnte sich beispielsweise ein zukünftiges Feld der interaktiven Bedienungsanleitung eröffnen. Im Internet finden sich bereits zahlreiche Anwendungen in dieser Richtung. Automobilhersteller bieten z.B. die Möglichkeit, mittels 3D-Konfiguratoren sein eigenes Auto zusammenzustellen. Ein weiterer Konfigurator erlaubt das Zusammenstellen eines Bürostuhls bei dem man einzelne Komponenten inklusive Farbe, Material usw. einpassen kann. WebGL erlaubt hierbei eine Anpassung in Echtzeit. Die Informationen des konfigurierten Produkts werden gespeichert und können z.B. als PDF heruntergeladen werden.

Für mich selbst war es interessant das Repertoire zu erforschen, das die 3D-Technologien im Internet um WebGL und z.B. X3D mit sich bringt. Besonders das riesige Aufgebot an Frameworks und anderen Vorlagen im Web ist enorm und ermöglicht eine Fülle an Kombinationsmöglichkeiten. Hier fand ich auch den Umfang des Programms Blender überraschend groß und gerade die Einbindungsfähigkeiten der unzählbaren zusätzlichen Features beeindruckend. Die Optionen, welche sich dabei für Transformationen im Web bieten, lassen kaum Wünsche offen.

Abbildungsverzeichnis