HoloEarth

Leonhard Onken-Menke, Sebastian Olariu

Inhalt

1 Einführung

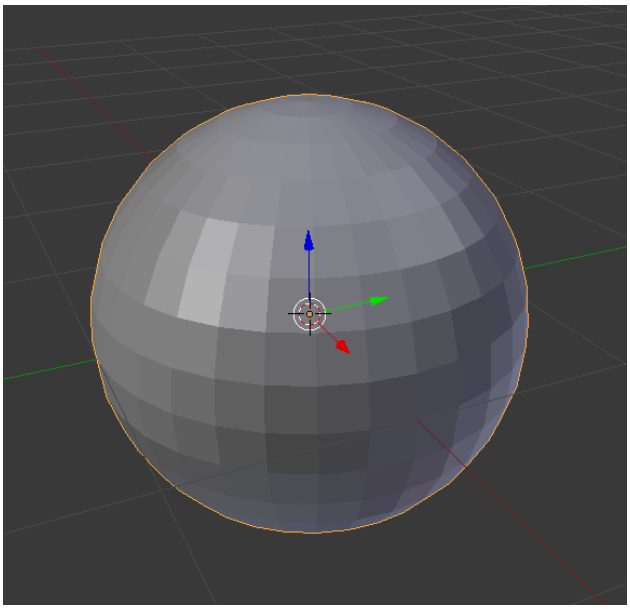
Die nachfolgende schriftliche Dokumentation beschäftigt sich mit der Erstellung einer kleineren Anwendung, in der der Nutzer eine virtuelle Weltkugel mit der HoloLens betrachten und entdecken kann. Die Weltkugel schwebt dabei im Raum und der Nutzer entdeckt die verschiedenen Kontinente, indem er diese „anklickt“. Dreidimensionale Kontinent-Platten visualisieren, welcher Kontinent angewählt ist. Durch Texttafeln, die neben dem jeweiligen Kontinent schweben, werden Informationen zum jeweiligen Kontinent angezeigt. Zudem dient die Weltkugel, dank hochaufgelöster Texturen, als virtueller Globus, den der Nutzer auf seine eigene Weise inspizieren kann.

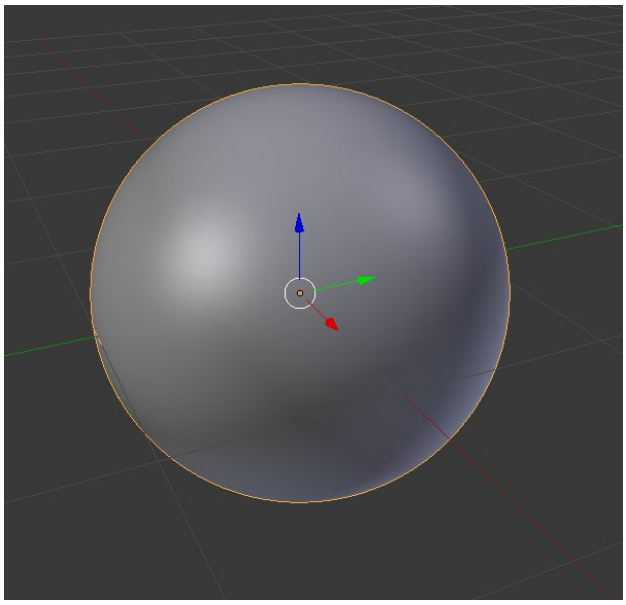
2 Blender

Für die Anwendung sind insgesamt acht Modelle nötig, ein Modell für die Erdkugel und siebe für die einzelnen Kontinente.

2.1 Erdkugel

Die Erdkugel ist eine einfache UV-Sphere, also eine Kugel. Da Blender Kugeln erstmal nicht ganz runde darstellt, wurde die Kugel mit mehr Flächen versehen und zusätzlich wurde sie geglättet.





2.2 Kontinente

Im Grunde bestehen die Kontinente aus jeweils einer Plane, also einer einfachen Fläche. Diese wurden so gerichtet, dass sie direkt vor der Kugel liegen. Zwischen diesen Flächen und der Kugel wurde die Textur der Weltkarte gelegt und die Fläche wurde halb durchsichtig gemacht. Anschließen wurde aus den Flächen, mit dem sog. Knife-Tool, die jeweiligen Kontinente nachgefahren und so ausgeschnitten. Damit die Flächen nun dreidimensional werden, wurden sie extrudiert, sie wurden also „breitgezogen“.

Zum Schluss hat jeder Kontinent noch seine eigenes Material bekommen, welches aus jeweils einer anderen, einfachen Farbe bestand.

3 Texttafeln

Die Texttafeln wurden in Photoshop designet, mit Informationen von Wikipedia bestückt und als PNG-Datei exportiert.

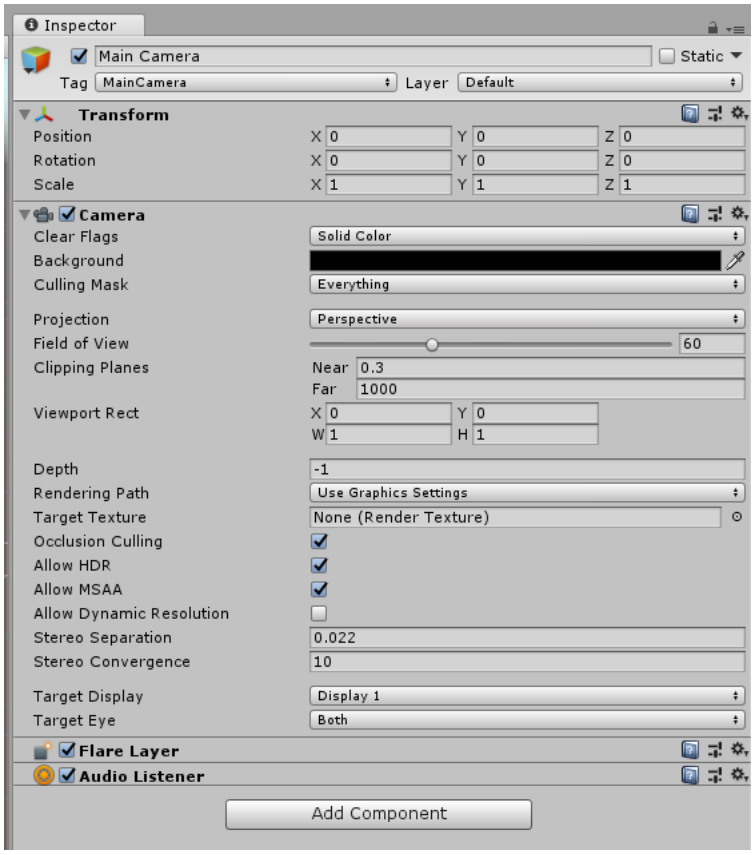
4 Unity

Für die Umsetzung dieses Projekts wurde Unity in der Version 2018.1.6f1 verwendet.

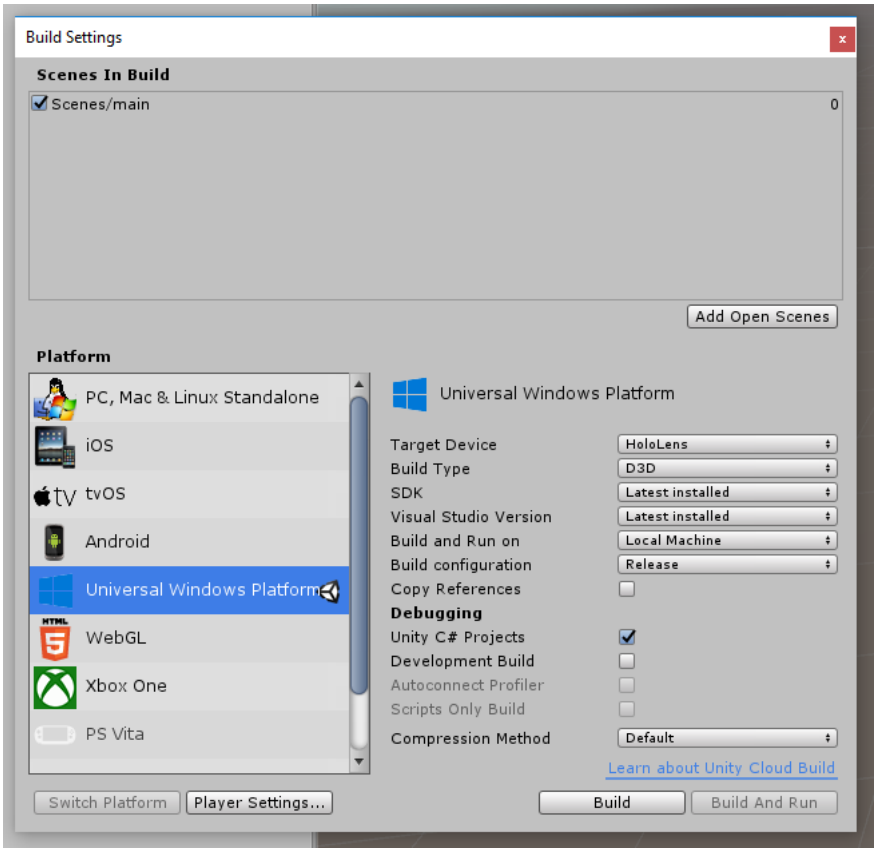
4.1 Voreinstellungen

Damit das Projekt auf der HoloLens funktioniert, müssen einige Voreinstellungen in Unity getroffen werden:

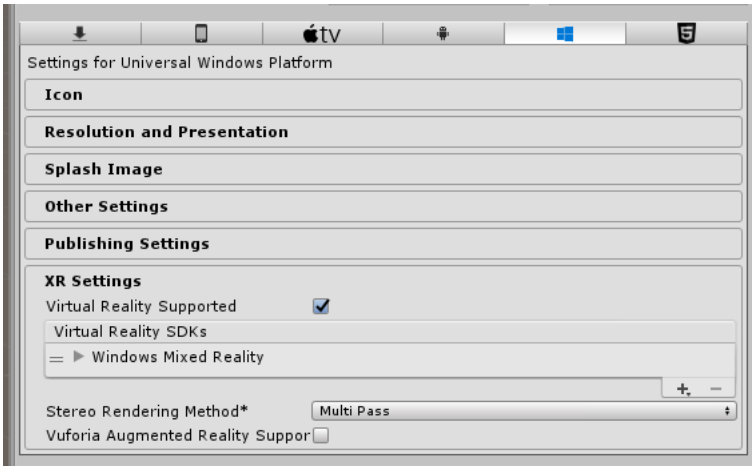
- Die Position der Kamera sollte auf (0, 0, 0) liegen, da die Position nämlich die des Nutzers entspricht. Von dieser Position aus wird auch alles berechnet.
  Außerdem müssen die Kamera auf **Solid Color**, und der Hintergrund (Background) auf schwarz gestellt werden. Für die HoloLens bedeutet der schwarze Hintergrund nämlich, dass sie dort nichts anzeigen soll und entsprechend kann man so die Umgebung sehen.

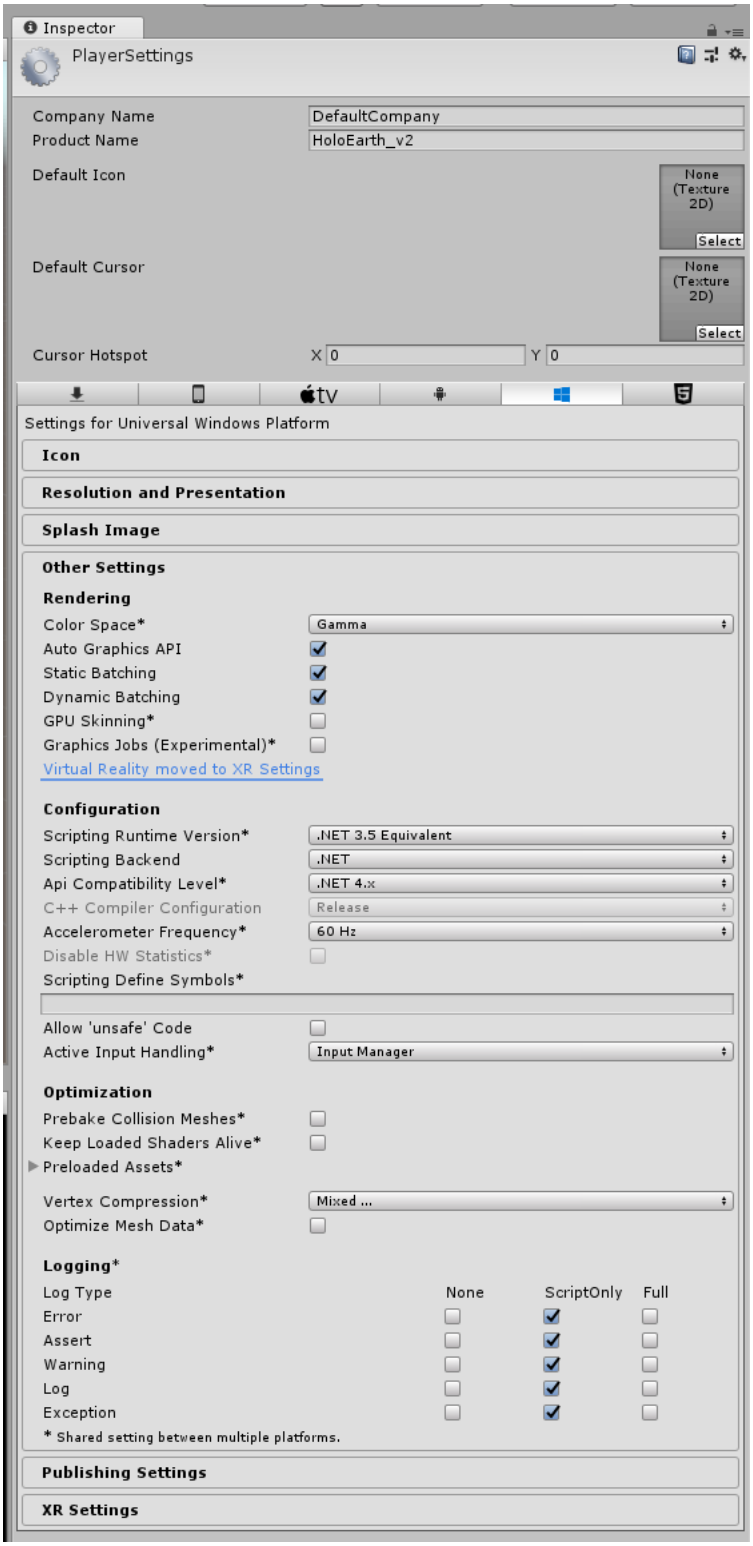


- In den **Build Settings** (File > Build Settings…) muss die Plattform auf **Universal Windows Platform** gestellt werden, die restlichen Einstellungen wurden wie auf dem folgenden Screenshot gewählt:



- Als letztes müssen noch zwei Einstellungen in den Player Settings vorgenommen werden.
  Zum einen muss hier unter den XR Settings ein Hacken bei **Virtual Reality Supported** gesetzt werden und unter Other Settings muss das Scripting Backend auf **.NET** gesetzt werden.





4.2 Plug-In

Damit die Anwendung auf der HoloLens funktionieren kann, muss das Unity Projekt um das HoloLens Toolkit erweitert werden. Für dieses Projekt wurde die Version „HoloToolkit-Unity-2017.4.0.0“ verwendet

4.3 Modell und Texturen importieren und positionieren

Das exportierte Blender Modell wurde - genau wie die Texturen der einzelnen Texttafeln und des Globus - wurde zuerst per Drag and Drop in das Unity Projekt gezogen. Anschließend wurde das Modell in die Szene gezogen und 1,5 Einheiten vom Ursprung entfernt platziert (wird das Modell zu nah an die Kamera platziert, wird es nicht angezeigt).

Für die Texttafeln wurden einfache Würfel erstellt, von der Größe her angepasst und neben den Kontinent Platten platziert. Die Texturen wurden dann einfach per Drag an Drop auf die jeweiligen Quader gezogen. Das Gleiche wurde auch mit der Textur für die Weltkugel gemacht.

Jede der Texttafeln haben eigene Spotlights als Lichtquellen bekommen.

4.4 Skript

Im Grunde ist das Skript in zwei Teile aufgeteilt.

Der erste Teil besteht aus mehreren solcher Blöcke:

```
1     public void EuropaAn () {
2
3             GameObject europa = GameObject.Find("Europa");
4             this.meshRenderer = europa.GetComponent<MeshRenderer>();
5             this.meshRenderer.enabled = true;
6
7             GameObject europaTafel = GameObject.Find("EuropaTafel");
8             this.meshRenderer = europaTafel.GetComponent<MeshRenderer>();
9             this.meshRenderer.enabled = true;
10
11            AfrikaAus ();
12            AntarktisAus ();
13            AsienAus ();
14            AustralienAus ();
15            NordamerikaAus ();
16            SuedamerikaAus ();
17
18    }
19
20    void EuropaAus () {
21
22            GameObject europa = GameObject.Find("Europa");
23            this.meshRenderer = europa.GetComponent<MeshRenderer>();
24            this.meshRenderer.enabled = false;
25
26            GameObject europaTafel = GameObject.Find("EuropaTafel");
27            this.meshRenderer = europaTafel.GetComponent<MeshRenderer>();
28            this.meshRenderer.enabled = false;
29
30    }
```

Für jeden Kontinent gibt es jeweils einen solchen Block, welcher aus zwei Funktionen besteht.

Die erste Funktion - `public void KontinentnameAn ()` - ist dafür zuständig, dass die Kontinent-Platten und die Texttafeln der jeweiligen Kontinente sichtbar, und die der anderen unsichtbar gemacht werden.

In obigen Beispiel wird in Zeile 3 nach dem Objekt „Europa“ gesucht (so heißt die Kontinent-Platte) und einem neuen Objekt zugeteilt. In Zeile 4 wird dann aus dem Objekt der sog. „Mesh Renderer“ genommen (der ist dafür zuständig das Objekt zu erzeugen bzw. darzustellen; in diesem Projekt ist er erstmal deaktiviert, damit am Anfang nur die Erdkugel, und nicht auch schon die Kontinent-Platten und Texttafeln, sichtbar ist) und in Zeile 5 wird der Mesh-Renderer dann auf *true* gesetzt, er wird also aktiviert.

Dasselbe passiert auch in den Zeilen 7-9 für die Texttafel.

Am Ende der Funktion werden dann noch die jeweiligen `void KontinentnameAus ()` - Funktionen der anderen Kontinente ausgeführt.

Die zweite Funkion - `void KontinentnameAus ()` - macht genau dasselbe wie der obere Teil der anderen Funktion, nur, dass die Mesh-Renderer hier auf *false* gesetzt werden, sie werden also deaktiviert.

Der zweite Teil des Skripts befindet sich in der Update-Methode:

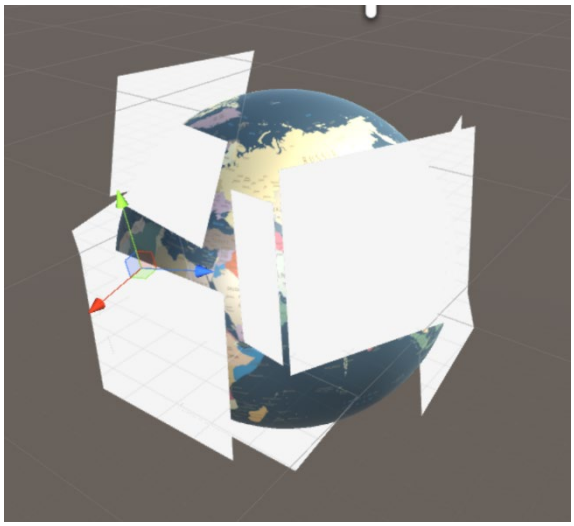
```
1     void Update () {
2
3           if (Input.GetKeyDown(KeyCode.A)) {
4
5                 AfrikaAn ();
6
7           }
8
9           if (Input.GetKeyDown(KeyCode.B)) {
10
11                AntarktisAn ();
12
13          }
14
15          if (Input.GetKeyDown(KeyCode.C)) {
16
17                AsienAn ();
18
19          }
20
21          [...]
22
23    }
```

Dieser Teil ist für die Anwendung auf der HoloLens irrelevant und nur zu Testzwecken interessant. Im Grunde werden hier nur verschiedene Tasten ausgelesen und so die Funktionen des anderen Teils aufgerufen.

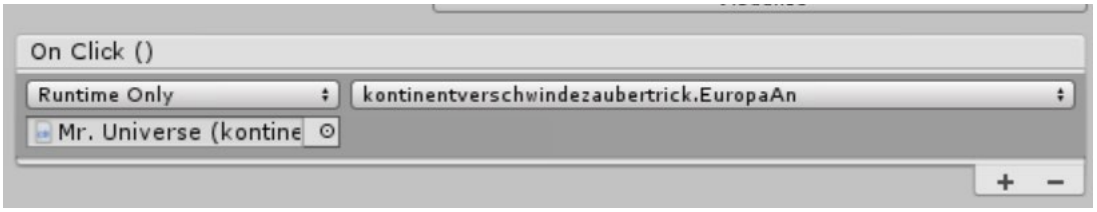
4.5 Buttons

„Anklickbar“ werden die Kontinente über sog. UI-Buttons (UI steht hier für *User Interface*).

Die Buttons sind in den beiden folgenden Bildern zu erkennen:



Im folgenden Ausschnitt ist zu sehen, wie die Buttons mit dem Skript und den passenden Funktionen verbunden sind:

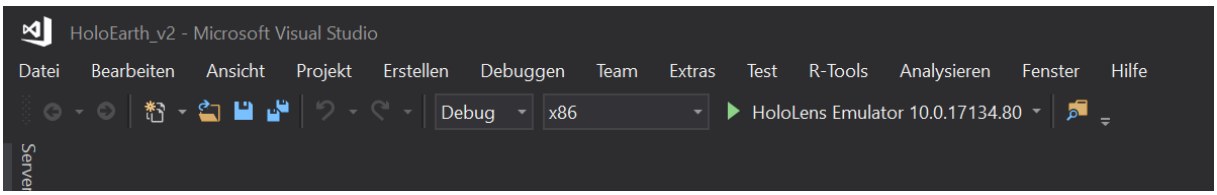


5 Visual Studio/HoloLens

5.1 Test auf dem Emulator

Bevor die Anwendung auf der HoloLens getestet wurde, wurde sie mit dem HoloLens Emulator - einer Erweiterung für Visual Studio *(https://docs.microsoft.com/en-us/windows/mixed-reality/install-the-tools)* - getestet.

Um die Anwendung mit dem Emulator zu testen, muss Visual Studio wie folgt eingestellt werden:



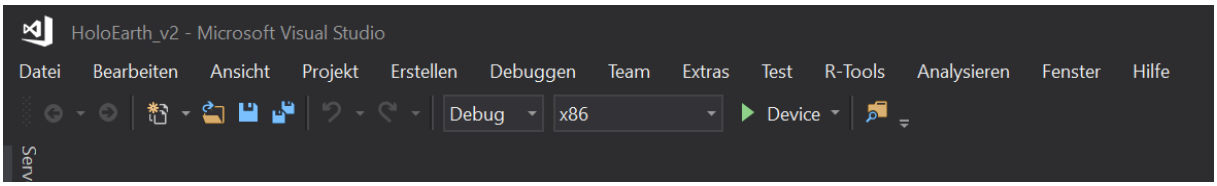
Gestartet wird der Emulator über einen Klick auf **HoloLens Emulator …** (nicht wundern, es könnte etwas dauern, bis die Anwendung gestartet ist)

Der laufende Emulator sieht dann so aus:

5.2 Übertragung der Anwendung auf die Brille

Im letzten Schritt wurde die Anwendung per USB auf die HoloLens übertragen.

Ist die Brille per USB-Kabel an den Rechner angeschlossen, kann die Anwendung über Visual Studio auf die Brille übertragen werden. Dazu muss Visual Studio wie folgt eingestellt werden:



Gestartet wird die Übertragung mit einem Klick auf **Device**. Die Anwendung ist anschließend startklar und kann sofort auf der HoloLens getestet werden.