


Handheld AR-Anwendung mit Vuforia und Unity3D

In dieser Übung erstellen wir eine *Handheld AR*-Anwendung Vuforia und Unity3D.

Vuforia erkennt vordefinierte Tracking-Marken und erlaubt die geometrische Registrierung anhand dieser. Wir wollen die mit einer Smartphone- oder Tabletkamera oder Webcam gefilmte Realität mit einem Modell eines Motorblocks augmentieren.

Die beschriebene Anwendung lässt sich unter Windows und iOS problemlos entwickeln. Sogar unter Linux hatte ich Erfolg mit einer Unity-Installation in einer Windows 10-VirtualBox VM. Hier ließ sich sogar das Kamerabild der Webcam nutzen.

Aufgabe 1 (Neuen Vuforia-Account einrichten)

- a) Um das Vuforia-Framework in Unity verwenden zu können, brauchen Sie zunächst einen Entwickler-Account (kostenloser *Developer Account*). Gehen Sie dazu auf die Entwickler-Website von Vuforia und legen sich einen neuen Account an (<https://developer.vuforia.com> .
- b) Nachdem Sie sich angemeldet haben müssen Sie ein neues Projekt mit gültigem Lizenzschlüssel anlegen. Wählen Sie den Reiter „Develop“ → „License Manager“ und drücken Sie auf den Knopf „Get Basic“.
- c) Vergeben Sie einen Namen für Ihr Projekt. Bestätigen Sie dann, dass Sie mit den Bedingungen einverstanden sind.
- d) Wenn Sie jetzt auf das neu angelegte Projekt im „License Manager“ klicken, können Sie den Lizenzschlüssel einsehen. Diesen brauchen Sie später, wenn Sie Vuforia in Unity nutzen wollen.
- e) Damit Sie das im Rahmen des Projektes genutzte Tracking-Target nutzen können, müssen Sie zunächst eine neue Target-Datenbank anlegen.
- f) Gehen Sie auf den Reiter „TargetManager“ und drücken Sie auf den Knopf „Add Database“. Vergeben Sie einen Namen (z.B. THI) und wählen Sie als Typ die Option „Device“.
- g) Laden Sie sich das auf Moodle bereitgestellte Bild `EngineMarker.png` herunter.
- h) Klicken Sie auf die neu angelegte Datenbank und dann auf die Schaltfläche „Add Target“.
- i) Wählen Sie die Option „Single Image“.
- j) Laden Sie das von Bild `EngineMarker.png` als Datei hoch und geben Sie die Breite mit 0.2 (entsprechend 20 cm) an.
- k) Drücken Sie dann auf „Add“, um das neue Tracking-Target zur Datenbank hinzuzufügen.


Wie gut sich ein Bild als Marker eignet, wird mit einer Stern-Bewertung angezeigt. Der Motor ist ein sehr guter Marker. Die erkannten Features (markante Punkte in der Erkennung) kann man bei Interesse durch Klicken auf das Target in der Target-Datenbank betrachten (Hyperlink „Show Features“).



- l) Im „Target Manager“ müssen Sie abschließend noch auf die Schaltfläche „Download Database (All)“ drücken. Wählen Sie im folgenden Dialog die Option „Unity Editor“ und drücken Sie auf „Download“.
- m) Später müssen Sie das heruntergeladene Unitypackage in Unity importieren, um das neu angelegte Tracking-Target nutzen zu können.

Aufgabe 2 (Neues Unity-Projekt einrichten)

- a) Legen Sie ein neues Unity-2019.4-Projekt an. Installieren Sie über *Window* → *Package Manager* das Paket „Vuforia Engine AR“. Im *GameObject*-Menü finden Sie nun das Untermenü *Vuforia Engine*.
 - b) Erzeugen Sie über dieses Untermenü eine „AR Camera“.
- Nach dem Einfügen der *AR Camera* benötigen wir *Main Camera* nicht mehr. Löschen Sie letztere daher.
- c) Importieren Sie das aus von Vuforia heruntergeladene Unitypackage für das Tracking-Target (s. Ende der ersten Aufgabe).
 - d) Wählen Sie nun das Objekt „ARCamera“ im Hierarchy-Fenster aus und drücken Sie im Inspektor auf „Open Vuforia Engine configuration“.

Fügen Sie in der Textbox für „App License Key“ Ihren Vuforia-Lizenzschlüssel (ein ellenlanger Mischmasch von Groß-/Kleinbuchstaben, Ziffern und Vorwärtsschrägstrichen) von der Vuforia-Webseite  ein.

Aufgabe 3 (Vuforia Tracking-Target einfügen)

- a) Um das von Ihnen angelegte Tracking-Target in Unity verwenden zu können, müssen Sie zunächst das Vuforia-Prefab „ImageTarget“ zur Szene hinzufügen: Menü „GameObject→Vuforia Engine→Image“.

An der Stelle des Image-Targets wird das Bild des Markers angezeigt.

- b) Laden Sie jetzt das 3D-Modell *V-Engine.fbx* von Moodle herunter und importieren Sie das 3D-Modell in Unity: Einfach das FBX-Objekt per Drag and Drop in den Asset-Ordner von Unity

ziehen.

- c) Fügen Sie das importierte Modell des Motorblocks als Kindelement zum „*ImageTarget*“ hinzu (Drag and Drop auf das „*ImageTarget*“ im Hierarchy-Fenster).
- d) Selektieren Sie den Motor im Hierarchy-Fenster und skalieren Sie die Größe des 3D-Modells im Inspektor auf 10% herunter.
- e) Verschieben Sie den Motorblock etwas in Y-Richtung, so dass er über dem Tracking-Target liegt.

Hat man einen Rechner mit Webcam, so kann man durch Drücken des *Play*-Buttons die Anwendung testen. Man hält den ausgedruckten Marker (zu Testzwecken kann der Marker auch auf einem Tablet oder Smartphone angezeigt werden) vor die Webcam, daraufhin sollte im Kamerabild der Motor über dem Marker schwebend dargestellt werden.

Aufgabe 4 (Für Android-Tablet/Smartphone builden)

Wenn Sie wollen, können Sie die Anwendung für Android-Tablet oder Smartphone erstellen, die Schritte hierzu sind analog zu der Erstellung von Google Cardboard-Anwendungen. Nehmen Sie dazu die Einstellungen vor, die auf Blatt 2, Aufgabe 5 beschrieben wurden.

Legen Sie den ausgedruckten Tracking-Marker auf den Tisch und filmen Sie diesen mit der Tablet/Smartphone-Kamera. Über dem Trackingmarker sollte jetzt das 3D-Modell des Motorblocks erscheinen.

Aufgabe 5 (Hinzufügen von AR-Schaltflächen)

Im nächsten Schritt sollen die auf dem Tracking-Target aufgedruckten Knöpfe mit einer Funktion belegt werden.

- a) Fügen Sie einen virtuellen Knopf hinzu, indem Sie im Inspector des „*ImageTarget*“ die „*Advanced*“ Option des „*Image Target Behaviour*“-Skripts öffnen und dort „*Add Virtual Button*“ drücken.
Dies legt ein *VirtualButton*-Objekt als Kind von *ImageTarget* an.
- b) Verschieben und skalieren Sie den virtuellen Knopf solange, bis er ungefähr mit der Fläche des Plus-Symbols auf dem Tracking-Marker übereinstimmt. Besser, Sie machen ihn etwas kleiner, damit Sie ihn nachher problemlos mit einem Finger drücken können.
- c) Selektieren Sie den virtuellen Knopf und geben Sie im Inspector unter „*Virtual Button Behaviour*“ für den Namen „*grow*“ ein.
- d) Erstellen Sie eine weitere Schaltfläche für das Minus (nach Schema a) bis c)) und geben Sie ihr den Namen „*shrink*“.
- e) Erstellen Sie schließlich eine Schaltfläche für den Stern und geben Sie dafür den Namen „*explosion*“ an.

Aufgabe 6 (Skalierung)

Um den Motorblock mithilfe der virtuellen Schaltflächen skalieren zu können, muss das Skript „VB Engine Event Handler“ ergänzt werden.

- a) Laden Sie das Skript `VBEngineEventHandler.cs` aus Moodle. Fügen Sie es mit Drag & Drop in die Assets ein.
- b) Fügen Sie das Skript mit „Add Component“ zu `ImageTarget` hinzu. In das Feld „Engine“ des Skripts ziehen Sie aus der Hierarchie das Modell des Motors (`VEngine`).
- c) Für die Skalierung werden im Skript die Methoden „scaleUp“ und „scaleDown“ verwendet.

Mit

```
engine.transform.localScale *= 2f;
```

wird der Motorblock beispielsweise um den Faktor 2 vergrößert. Nutzen Sie statt des festen Wertes aber den im Editor einstellbaren Wert `scaleFactor`.

- d) Für die Explosionsansicht (Methode „toggleExplosion“ werden die Kindobjekte des Motorblocks verschoben. Hierzu muss der Vektor zwischen Eltern- und Kindobjekt berechnet werden. Den normalisierten Verschiebungsvektor erhält man über die Methode

```
getDirection(Vector3 fromV, Vector3 toV)
```

mit der Position des Elternobjekts *fromV* und der Position des Kindobjekts *toV*.

Um alle Kindobjekte zu verschieben, muss die Verschiebung *rekursiv* erfolgen, d.h. die Methode „toggleExplosion“ ruft sich selbst auf. Dieser Aufruf muss in diesem Fall nach der Verschiebung des Kindobjekts erfolgen, welches beim rekursiven Aufruf das neue Elternobjekt darstellt, wie hier veranschaulicht:

```
private void toggleExplosion(Transform parent){  
    foreach(Transform child in parent.transform){  
        //move child object  
        toggleExplosion(child);  
    }  
}
```

Zum Toggeln benötigt man die Variable `exploded` vom Typ `bool`.