Cookbook für SSIUnity

Sebastian Erfurth und Michael Kaleja

27. August 2016

Inhaltsverzeichnis

1 Import des Package

Assets >> Import Package >> Custom Package... >> Package auswählen

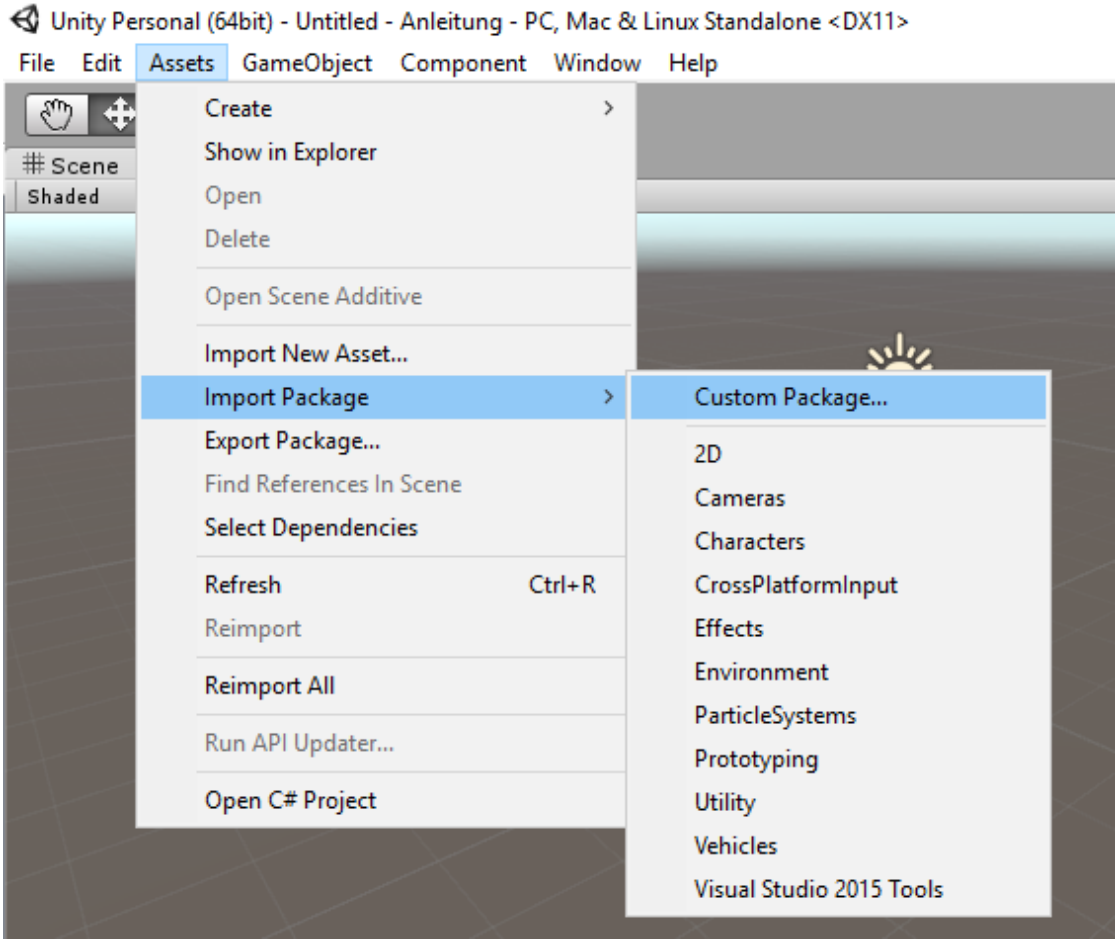

Abbildung 1.1: Import eines Package in Unity3D

oder

Bei geöffnetem Projekt im Explorer auf Package doppelt klicken.

Im darauffolgenden Dialogfenster kann der gewünschte Inhalt des Packages ausgewählt und anschließend importiert werden.

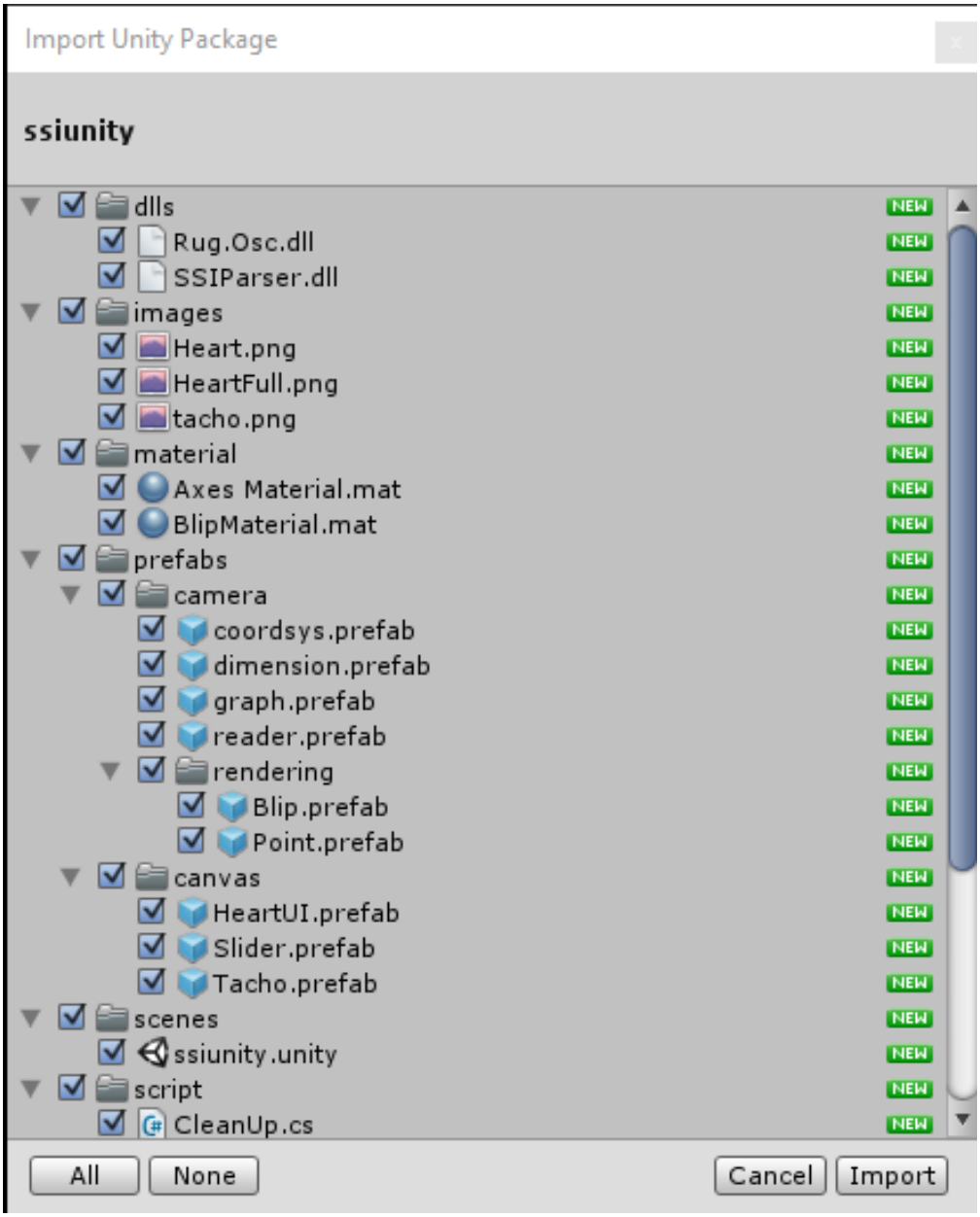

Abbildung 1.2: Auswahl von Package-Inhalt

Ein minimaler Import muss die dlls, die ReaderUtil und die SocketReader.xml enthalten. Zu den entsprechenden Prefabs müssen das dazugehörige Skript und ggf. die Materialien ausgewählt werden. Die Szene ssiUnity.unity enthält die beispielhafte Implementierung aller Prefabs.

2 Implementierung

Für die Implementierung der Prefabs liegen Skripte bei.

2.1 Spielobjekte

Jedem Spielobjekt liegt eine Kamera bei, welche für die Darstellung verantwortlich ist. Somit wird die Positionierung über die Kamera durchgeführt.

Graphen Graphen sind mit dem Skript SignalGraph ausgestattet. Es muss der Name des Kanals angegeben werden, auf den gehört werden soll. Der GraphOffset beschreibt den Abstand zwischen den Dimensionen des Graphen. Das heißt, bei zweidimensionalen Signalen werden zwei Graphen untereinander gezeichnet.

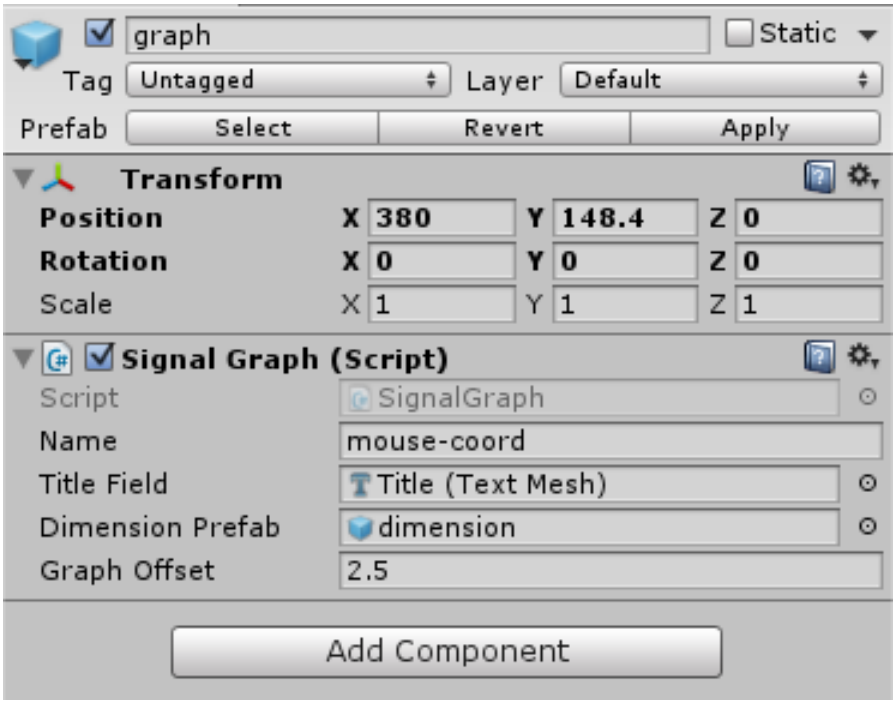

Abbildung 2.1: Auswahl von Package-Inhalt

Koordinatensystem Das Skript PositionGraph liegt im *drawer* des *coordsys*-Prefabs. Auch hier ist der Name für die Identifizierung des Kanals erforderlich. Mit *DisplayXEnd* und *DisplayYEnd* können die Höhe und Breite des Koordinatensystems festgelegt werden.

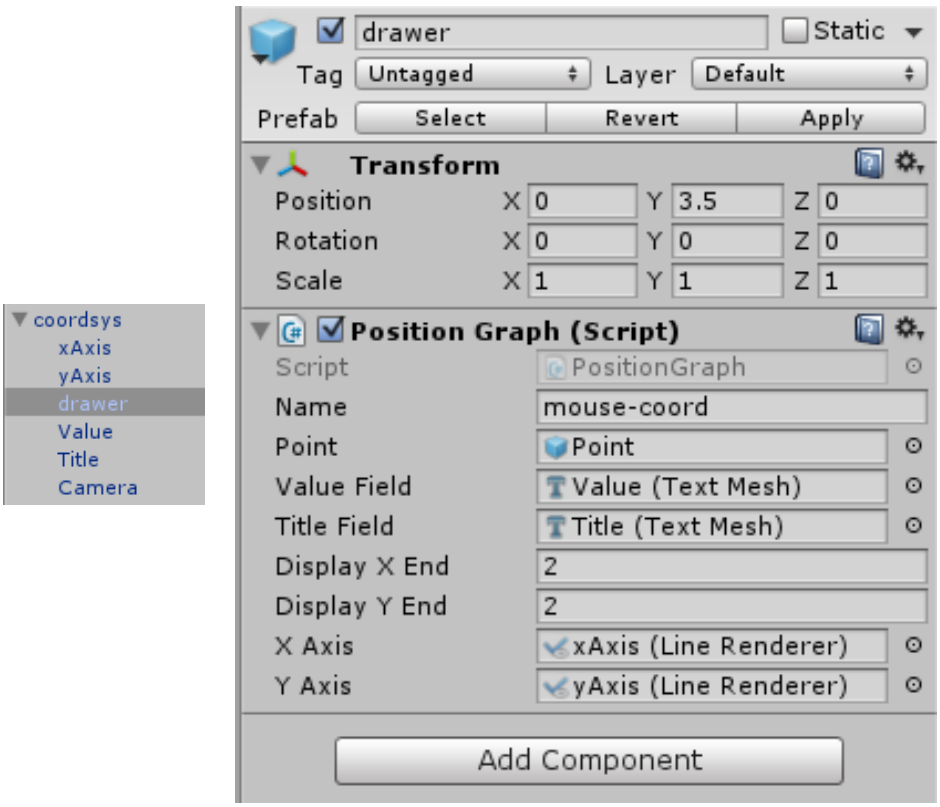

Abbildung 2.2: Skript für Koordinatensystem

Reader Reader sind mit dem Skript GraphLoader ausgestattet. Es muss der Name der Kanal-Gruppe angegeben werden, welche angezeigt werden sollen. Der Offset beschreibt den Abstand zwischen den Graphen der Gruppe.

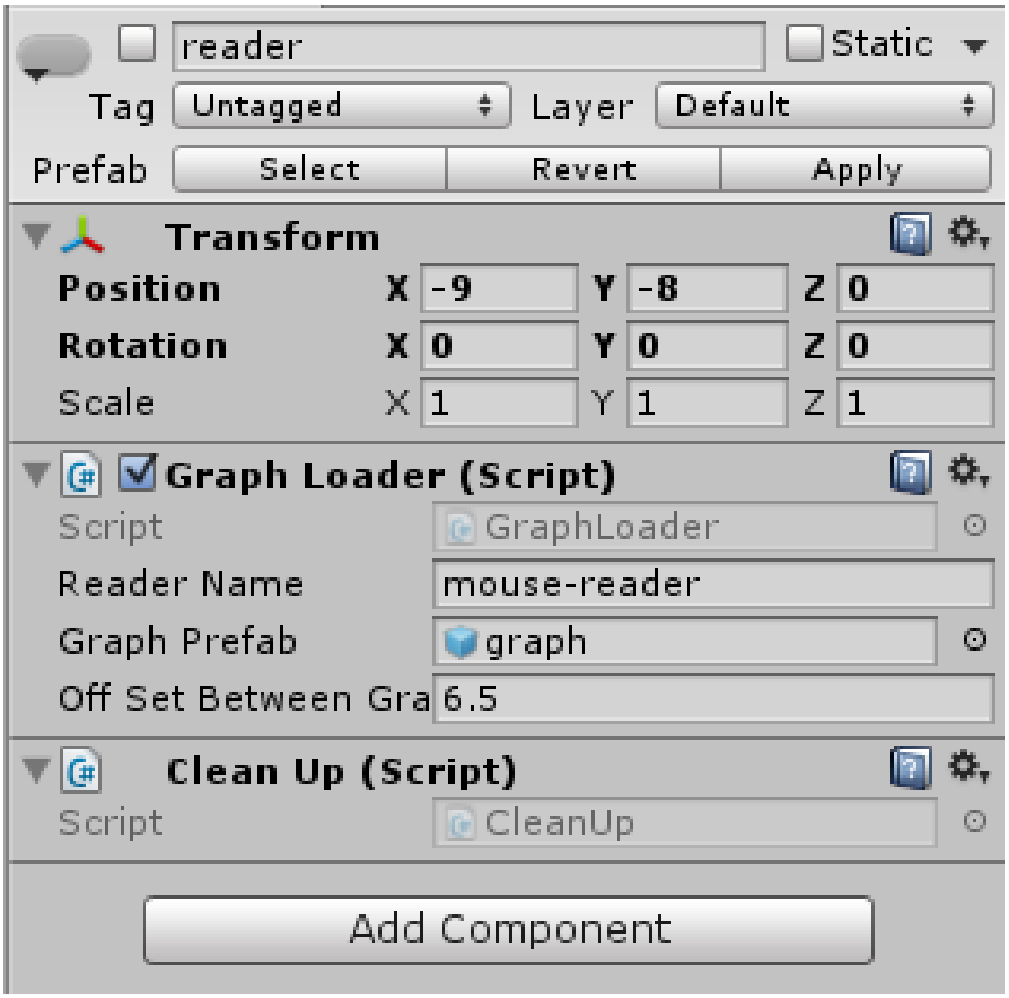

Abbildung 2.3: Auswahl von Package-Inhalt

2.2 UI-Elemente

Um die Werte in den UI-Elementen zu setzten, muss die *SetValue*-Methode der entsprechenden Skripte aufgerufen werden.

Schieberegler Das Skript *SliderManager* befindet sich im Slider selbst. Zusätzlich zur *SetCurrentValue*-Methode kann der Wert auch mit der *IncreaseCurrentValue*-Methode angepasst werden.

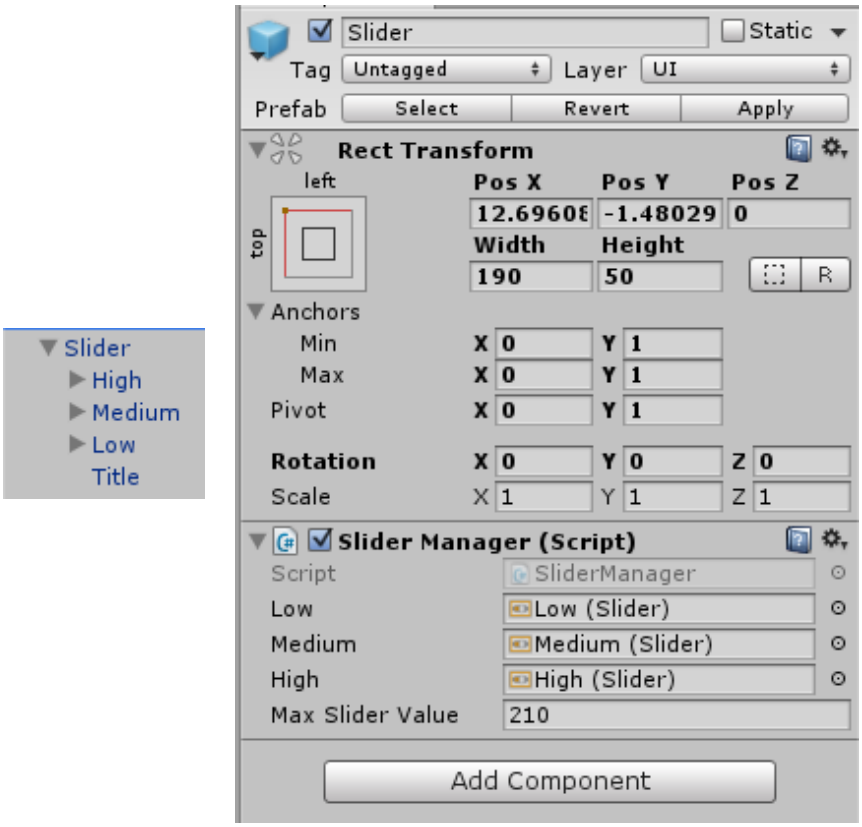

Abbildung 2.4: Skript für Slider

Wenn die Werte des Sliders angepasst werden sollen, müssen die Maximalwerte der drei inneren Slider Low, Medium und High justiert werden, wie folgende Abbildung zeigt:

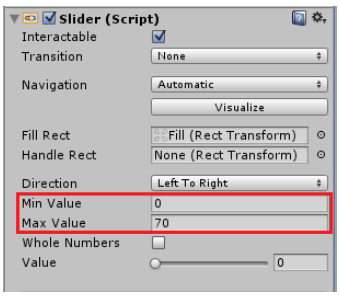

Abbildung 2.5: Maximalwert für Sliders anpassen

HeartUI Das Skript HeartBeat befindet sich im Bild HeartFull. Bevor der Wert über die *SetValue*-Methode angepasst werden kann, muss die Framerate über die *SetFrameRate*-Methode gesetzt werden.

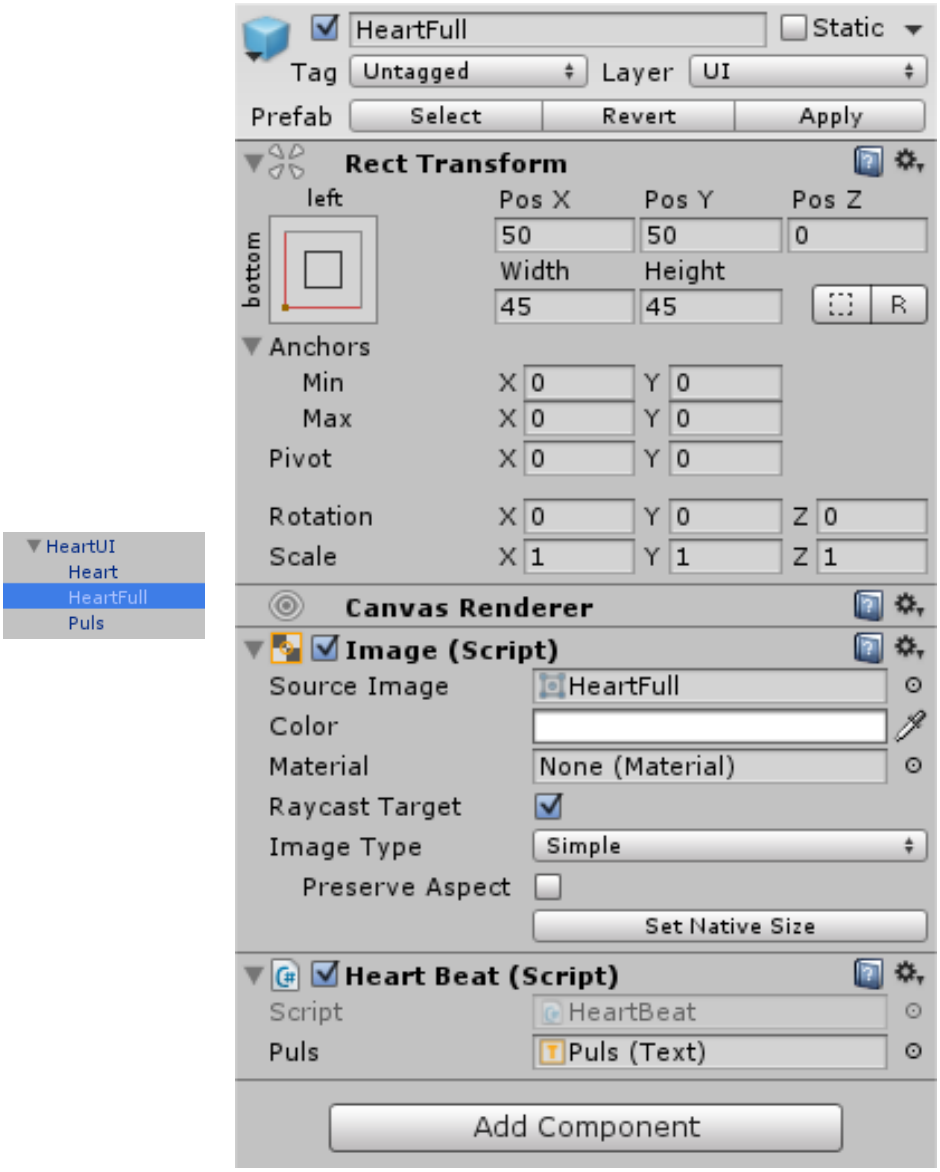

Abbildung 2.6: Skript für die HeartUI

Tachometer Das Skript TachoManager befindet sich im Needle-Objekt des Prefabs Tacho. Es kann die Aktualisierungsrate der Nadel durch das Setzen des Wertes von RotationUpdateTime gesetzt werden. Empfohlen wird der Wert null. Für eine fehlerfreie Darstellung, muss der über die *SetValue*-Methode gesetzte Wert zwischen null und eins normalisiert sein.

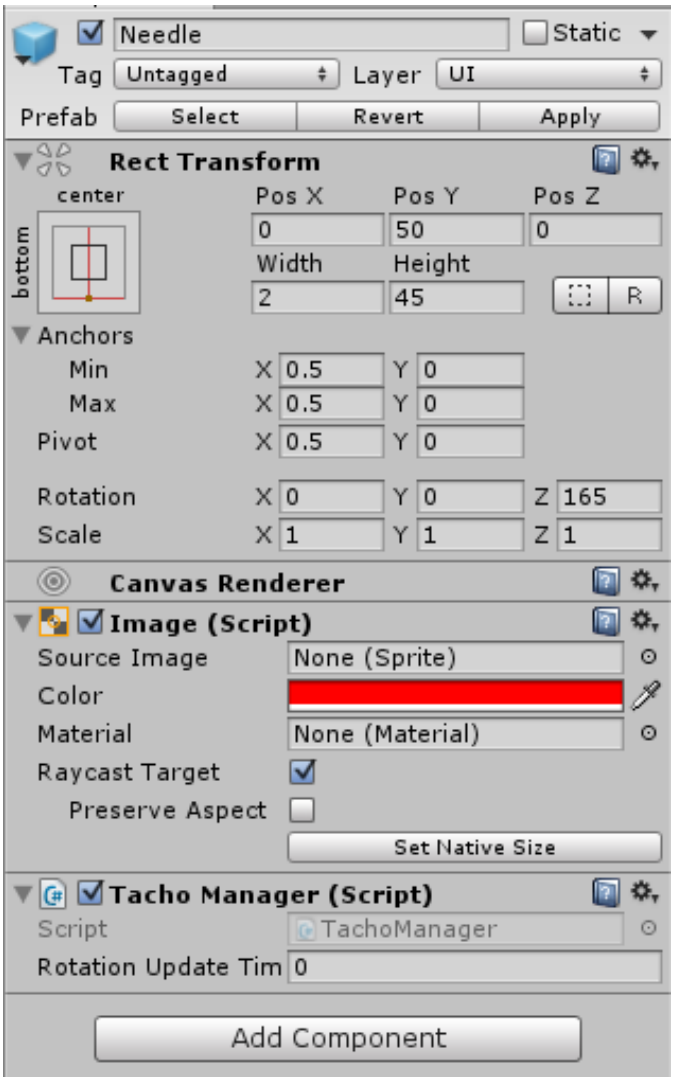

Abbildung 2.7: Skript für Tachometer

2.3 SocketReader.xml

Die SocketReader.xml ist für die Konfiguration der Kanäle zuständig. Sie ist dem Unity3D-Package enthalten und muss an die eigenen Bedürfnisse angepasst werden. Der Port muss dem vom Socket-Writer seitens SSI entsprechen. Die Dimension, Datentyp und Samplerate muss der SSI-Dokumentation entnommen werden.

```
<?xml version="1.0" encoding="utf-8" ?>
<readers xmlns="http://tempuri.org/Reader.xsd">
        <channelgroup name="Mouse" id="1" ip="127.0.0.1">
                <channel name="mouse-coord" port="8888" dim="2"
                        bytes="4" type="Float" rate="50" />
                <eventChannel name="click-event" port="8889"/>
        </channelgroup>
</readers>
```

Listing 2.1: SocketReader.xml

2.4 ReaderUtil

Bei der ReaderUtil handelt es sich um eine C#-Klasse, welche das Adapter-Entwurfsmuster implementiert. Sie stellt statische Methoden der ReaderStarter zum Starten, Stoppen und Finden der Kanäle bereit.

2.4.1 Streams

Anbei befindet sich ein Beispiel, wie anhand der ReaderUtil Stream-Channels abgefragt werden können:

```
/* Finden und zuweisen eines Kanals */
var socketReader = ReadersUtil.GetChannelByName("mouse-coord") as FloatChannel;

/* Abfrage der aktuellen Werte (1. Array: Position, 2. Array: Dimension) */
float[][] data = socketReader.GetCurrentData();

/* Der erste Wert der x und y Koordinate der Maus-Position */
float xCoord = data[0][0];
float yCoord = data[0][1];

/* Letzte Wert der x und y Koordinate der Maus-Position */
float[] lastData = socketReader.GetCurrentData();
xCoord = lastData[0];
yCoord = lastData[1];
```

Listing 2.2: Beispiel-Code für Aufrufe von ReaderUtil

Streams können durch einfachen Aufruf der entsprechenden Methoden abgefragt werden. Die Daten werden dabei automatisch aktualisiert sobald neue verfügbar sind. Im Regelfall ist dies abhängig von der Sample-Rate des Streams.

2.4.2 Events

Event-Channel sind nicht abhängig von einer Sample-Rate und müssen dementsprechend anders abgefragt bzw. implementiert werden.

Event-Channel basieren auf dem Beobachter-Entwurfsmuster. Hier muss zunächst von der entsprechenden Klasse bzw. dem Mono-Behaviour(Unity-Skript) die Schnittstelle Listener<OscPacket> implementiert werden:

```
public class PlayerShooting : MonoBehaviour, Listener<OscPacket>
{
        //...
        public void ReceiveEvent(Event<OscPacket> eventArg)
        {
                OscMessage message = eventArg.Content as OscMessage;
                string eventName = message[1] as string;
                if (eventName.Equals("air-exhale"))
                {
                        int duration = (int) message[3] / 100;
                        shellForce = duration;
                        fireAllowed = true;
                }
        }
        //...
}
```

Listing 2.3: Beispiel-Code Implementierung des Event-Listeners

Wie im Listing zu sehen kann nun über die ReceiveEvent-Methode das entsprechende Event abgefangen und ausgewertet werden. In diesem Fall werden für das Skript einige einfache Variablen gesetzt, wenn der Event-Name “air-exhale” ist. Der Aufbau des Osc-Pakets ist dabei abhängig vom Typ des SSI-Events. Der Typ muss bekannt sein, damit es beim Abfragen der Werte des Pakets zu keinen Fehlern kommt! Mehr zu den SSI-Events findet sich in der SSI-Dokumentation.

Es ist zu beachten, dass Unity3D den Zugriff auf Variablen nur im Main-Thread erlaubt. Aus diesem Grund ist sicherzustellen, dass alles, was in der ReceiveEvent-Methode geschieht thread-sicher sein muss! Das Schreiben von primitiven Typen funktioniert jedoch meist problemlos.

Nach der Implementierung der Schnittstelle muss zusätzlich der neue Beobachter auf den Event-Channel registriert werden. Hierfür bietet sich die ReadersUtil an, die den Channel per Port bzw. Name finden kann:

```
public class PlayerShooting : MonoBehaviour, Listener<OscPacket>
{
        //...
        private void Awake()
        {
                var airExhale = ReadersUtil.
                        GetChannelByName("air-event") as EventChannel;
                airExhale.AddListener(this);
        }
        //...
}
```

Listing 2.4: Beispiel-Code Registrierung Event-Listeners