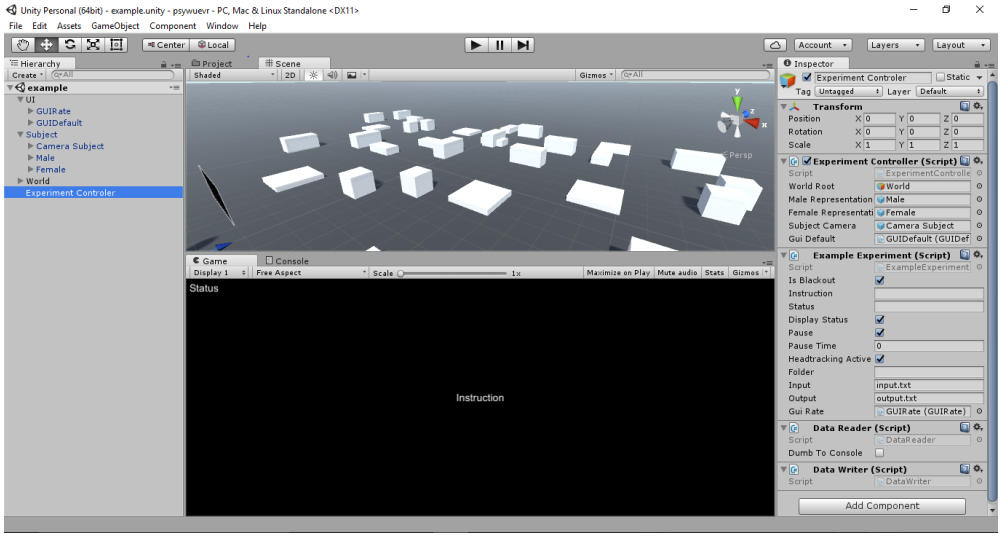

PsyWueVR
-Handbuch-

Nico Balbach

15. November 2016

Inhaltsverzeichnis

1 Einleitung

Das Projekt PsyWueVR ist ein Framework welches das Erstellen von psychologischen Virtual Reality Experimenten vereinfachen soll. Hierzu wird Programmcode, zum Arbeiten mit Unity und Oculus Rift zur Verfügung gestellt. Die Entwicklung erfolgte, im Auftrag des Psychologischem Lehrstuhl 1, von Nico Balbach.

2 Installation

Die Installation des Framworks geschieht in wenigen einfachen Schritten.

1. Lade das Unity-Package „PsyWueVR.unitypackage“ runter
2. Öffne das Unity-Projekt in dem es integriert werden soll oder erstelle ein neues Projekt
3. Öffne „PsyWueVR.unitypackage“ per Doppelklick oder Rechtsklick → „Öffnen“
4. Wähle „PsyWueVR“ und optional „PsyWueVR-Example“ für ein Beispielexperiment aus (Siehe 2.1)
5. Klicke „Import“ und warte bis der Prozess zu Ende ist

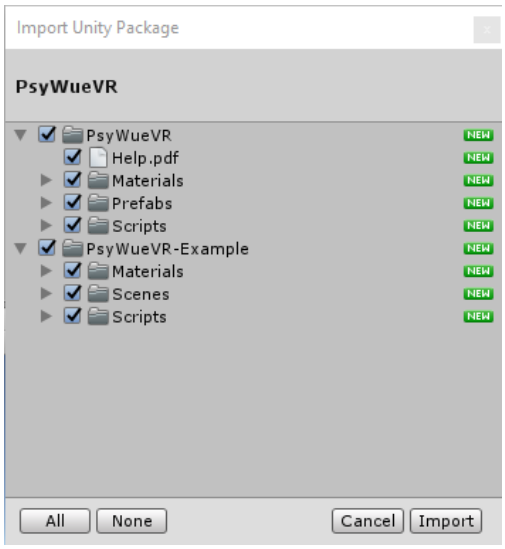

Abbildung 2.1: Import Framework

3 Aufbau

In diesem Kapitel wird der allgemeine Aufbau eines VR Experimentes mit „PsyWue-VR“ erklärt. Es können hierzu voreingestellte Objekte unter „PsyWueVR/Prefabs“ gefunden werden.

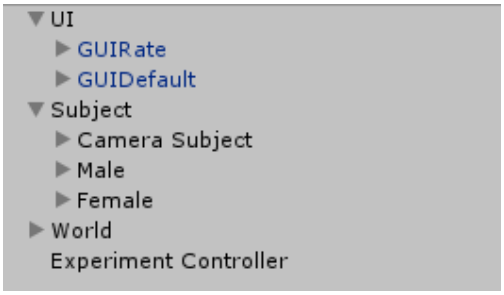

Abbildung 3.1: Projekt Struktur

3.1 Experiment Controller

Der Experiment Controller ist zur Hauptverwaltung der Experimente zuständig. Es enthält das Hauptskript „Experiment Controller“, die Hilfsskripte „Data Reader“, „Data Writer’ und eine Experiment Skript.

3.1.1 Experiment Controller (Skript)

Dieses Skript ist zur Hauptverwaltung zuständig und sollte im Normalfall nicht bearbeitet werden. Es nimmt ein Weltobjekt, die Repräsentation für Mann und Frau (siehe 3.2), sowie eine Kamera und das Standard GUI Objekt (weiteres in 3.3) entgegen.

3.1.2 Data Reader

Der Data Reader wird zum Auslesen einer Experiment-Text Datei benötigt (siehe 4.2). In einer solchen Textdatei können verschiedene Optionen für das Experiment, wie das Geschlecht des Teilnehmers, festgelegt werden.

3.1.3 Data Writer

Mit dem Data Writer können Daten des Experimentes leicht abgespeichert werden. Weiteres hierzu kann unter Kapitel 4.4 nachgelesen werden.

3.1.4 Experiment Skript

Für das individuelle Verhalten verschiedener Experimente benötigt der Controller ein Experiment Skript. Ein solches Skript muss von der Subklasse Experiment sein. Jedes Experiment beinhaltet Funktionen für Blackout, den Status/ die Instruction Texte, Pause, Headtreacking und die Namen der Input und Output Dateien. Weitere Informationen über den Umgang mit einem solchen Skript, siehe 4.1.

3.2 Subject

Ein Subjekt beschreibt die Darstellung eines Versuchsteilnehmers. Für jedes Experiment wird von zwei Objekten (männlich, weiblich) mit je einem „SubjectRepresentation“ Skript ausgegangen. Jedes Subjekt benötigt eine Darstellung und einen Kamera „Viewport“, welcher die Position der Kamera an diesem Subjekt darstellt (Beispiel Augenhöhe). Als Standard Darstellung wird ein Rocketbox-Avatar empfohlen. (Weiteres siehe 4.3)

3.2.1 SubjectRepresentation

Jedes Experiment benötigt ein Repräsentation-Skript. Ein solches Skript erbt von der Klasse „SubjectRepresentation“ und nimmt mindestens eine Teilnemerdarstellung und deren „Viewport“ (Kameraposition/Augenhöhe) entgegen. Gegebenenfalls können hier weitere für die Darstellung wichtige Objekte abgespeichert und im Experiment abgerufen werden. (Weiteres siehe 4.3.1)

3.3 UI

Die graphische Interface eines Experimentes wird durch das UI Objekt dargestellt. Dies ist ein CanvasObjekt, welches mehrere GUI Elemente mit einem GUIElement Skript beinhaltet.

3.3.1 GUIElement

Ein GUI ELement sollte von der Klasse „GUIElement“ erben. Sie beschreiben einzelne Interface Elemente, welche während eines Experimentes dargestellt werden sollen. Das Standartmäsige GUI Element ist GUIDefault, mit diesen können ein Statustext, sowie Anleitungstext dargestellt und ein Blackout erzeugt werden. Weitere Informationen zum Erstellen neuer GUI Elemente kann unter 4.5 nachgelesen werden.

4 Benutzung

In diesem Kapitel werden die programmiertechnischen Grundlagen zum Erstellen eines neuen Experimentes beschrieben. Zuerst werden die Grundlagen eines Experiment-Skriptes dargelegt. Daraufhin wird auf das Auslesen der Optionsdatei und das Ausgeben von Experimentdaten in eine Dateiausgabe eingegangen. Zum Schluss wird das Hinzufügen neuer GUI Elemente erläutert.

4.1 Experiment

Um ein Experiment zu erstellen, muss ein von „Experiment“ erbendes Skript erstellt werden. Die Klasse Experiment kann vom ExperimentController eingelesen werden und nimmt dem Programmierer einige Arbeit ab.
Ein Beispielexperiment kann mit folgenden Methoden aufgebaut werden (Ein vollständiges Beispielexperiment befindet sich unter „PsyWueVR-Example“).

```
1  public class ExampleExperiment : Experiment {
2
3    /* Hier werden Variablen initialisiert, welche vor dem Einlesen der
       Optionsdatei bestehen sollen. Bsp.: den Namen/Speicherort der
       Optionsdatei */
4    public override void initValues(){..}
5
6    /* Weitere Initialisierungen, welche u.a. Daten aus der Optionsdatei
       verwenden, sollten hier festgelegt werden.
7      z.B. Die einzelnen Phasen des Experimentes */
8    public override void init(){..}
9
10   /* In dieser Methode sollten Funktionen definiert werden, welche nach
       dem Experimentenstart und vor der ersten Experimentphase starten
       sollten*/
11   public override void startIndividual(){..}
12 }
```

(Die Methoden initValues() und startIndividual() sind optional)
Dem Experiment stehen einige Variablen zur Verfügung, darunter sind:

```
1    // Name des Experiments
2    protected string name;
3    // Startzeit des Experimentes
4    protected float startTime;
5    // Liste aller Experimentphasen
6    protected List<ExperimentPhase> phases;
7    // Nummer der aktuellen Experimentphase
8    protected int current;
```

```
9   // Boolean zum pausieren des Experimentes (das Setzen von pause = true,
        pausiert das Experiment)
10  public bool pause;
11
12  // Subjektdaten
13  public Subject subject;
14
15
16  // Default GUI (Erleichterte Verwendung mit 'isBlackout', 'instruction'
        und 'status')
17  public GUIDefault GUIDefault;
18  // Boolean zum Starten eines Blackouts (das Setzen von isBlackout =
      true, erzeugt einen Blackout/Schwarzen Bildschirm)
19  public bool isBlackout = true;
20  // Ein Mittiger Text um Instruktionen darzustellen. (Der Stringinhalt
      von instruction wird sofort auf dem Bildschirm dargestellt)
21  public string instruction;
22  // Ein an der linken oberen Ecke befindlicher Statustext (zum debuggen
      geeignet. Verhalten wie instruction)
23  public string status;
24  // Boolen um Status zu deaktivieren (displayStatus = true verschleiert
      den Statustext
25  public bool displayStatus;
26
27
28  // Boolean zum aktivieren und deaktivieren von Headtracking der
      Versuchsperson (headtrackingActive = true aktiviert das headtracking)
29  public bool headtrackingActive = true;
30
31  // DataWriter zum Beschreiben der Experimentdatei
32  public DataWriter writer;
33  // Ordner in dem sich die Inputdatei befindet und die Outputdatei
      erstellt werden soll
34  public string folder;
35  // Name der Inputdatei
36  public string input;
37  // Name der Outputdatei
38  public string output;
```

Weitere Variablen welche in der vererbten Experiment Klasse definiert sind, können unter „PsyWueVR/Scripts/Abstracts/Experiment“ nachgesehen werden.

4.1.1 Experimentphasen

Experimentphasen sind Phasen, welche nach dem Start des Experimentes nacheinander ausgeführt werden. Diese Phasen besitzen eine Dauer, sowie eine Methode zur Ausführung und werden in der Liste „phases“ gespeichert. Eine solche auszuführende Methode besitzt weder Übergabe- noch Rückgabewert. Die Phasen sollten in der Methode init festgelegt werden und könnten wie folgt aufgebaut werden:

```
1  public class ExampleExperiment : Experiment {
2
3    public override void init(){
4      // Starte Methode "'greeting"' und warte 10 Sekunden bis zur
       nachfolgenden Methode
```

```
5       phases.Add (new ExperimentPhase (10, greeting));
6       // Starte Methode "'blackout"' und warte 5 Sekunden
7       phases.Add (new ExperimentPhase (5, blackOut));
8       // Starte Methode "'end"' und warte 10 Sekunden bis das Experiment
        beendet wird (letzte Methode)
9       phases.Add (new ExperimentPhase (10, end));
10    }
11
12    // Phase ohne Blackout und einem Starttext
13    private void greeting ()
14    {
15      isBlackout = false;
16      status = "Greeting";
17      instruction = "Hallo!";
18    }
19
20    // Phase mit reinem Blackout
21    private void blackOut ()
22    {
23      isBlackout = true;
24      status = "Blackout";
25      instruction = "";
26    }
27
28    // Phase mit Blackout und Endnotiz
29    private void end ()
30    {
31      isBlackout = true;
32      status = "End";
33      instruction = "Dies ist das Ende.";
34    }
35  }
```

4.2 Optionsdatei

In der Optionsdatei können verschiedene Optionen für das Experiment festgelegt werden. Die zu öffnende Datei wird mit den Variablen „folder“ und „input“ bestimmt. Fall dort keine Datei vorhanden ist, wird nach dem ersten Experimentstart eine Dummy-Datei erstellt. Pro Zeile kann eine Variable festgelegt werden. Eine solche Variable besitzt einen Variablennamen und einen Wert. In einer Zeile wird dies durch „$< name >=< wert >$“ festgelegt. Das Symbol # beschreibt ein Kommentar. Eine Zeile die mit # beginnt wird nicht weiter eingelesen. Eine Optionsdatei kann wie folgt aufgebaut werden:

```
1 # Beispiel Input file
2 subject=01
3 gender=m
4 name=Max Mustermann
5 job=student
```

Die Variable „gender“ wird von jedem Experiment erwartet. Die Optionen sind „m“ für männlich und „f“ für weiblich.

4.3 Subject

Das Objekt „subject“ beinhaltet Informationen zur Versuchsperson. Darunter einer „SubjectRepresentation“ und Daten welche in der Optionsdatei festgelegt wurden. Auf die Daten der Optionsdatei kann mithilfe eines Dictionaries zugegriffen werden. Bsp.: Die Ausgabe des Geschlechts („f“ oder „m“) in die Konsole, kann wie folgt erzeugt werden.

```
print(subject.data ["gender"]);
```

Auf die Representation des Teilnehmers kann wie folgt zugegriffen werden:

```
subject.representation;
```

4.3.1 Teilnehmer Representation

Eine Teilnehmer Representation muss mit einem eigenen Skript definiert werden, welches von „SubjectRepresentation“ erbt. Solche Representationen können auf den Avatar („avatar“) und die Kamera des Teilnehmers („viewPoint“) zugreifen. Das eigene Skript kann daher auch zum Hinzufügen eigener Objekte genutzt werden. Falls der Teilnehmer z.B. GameObjects wie einen Stift und ein Blatt Papier besitzen soll, kann ein Skript wie folgt aufgebaut werden:

```
using UnityEngine;
using System.Collections;

public class ExampleRepresentation : SubjectRepresentation {
  public GameObject pen;
  public GameObject paper;
}
```

In einem Experiment kann auf diese nun unter anderem in einer Experimentenphase, wie folgt zugegriffen werden:

```
public class ExampleExperiment : Experiment {
  [..]
  private void phaseX ()
  {
    GameObject pen = subject.representation.pen;
    GameObject paper = subject.representation.paper;
  }
}
```

4.4 Dateiausgabe

Das Object „public DataWriter writer“ kann zum Abspeichern von Experimentergebnissen verwendet werden. Es wird beim Start des Experimentes automatisch ein solches Objekt erstellt. Standardmäßig wird darauf hin die Datei unter „folder+output“ (Variablen des Experimentes) geöffnet und nach dem Ende des Experimentes auch wieder geschlossen.
Der Writer besitzt folgende Methoden:

```
using UnityEngine;
using System.Collections;
using System.IO;

public class DataWriter : MonoBehaviour
{
[..]

  // Einlesen der Datei unter "'fileDirectory"'
  public void open (string fileDirectory){..}

  // Schliessen der offenen Datei
  public void close (){..}

  // Schreibe einen Header mit Experiment Namen "'experiment"'
  public void writeHeader (string experiment){..}

  // Schreibe einen Footer mit Experiment Namen "'experiment"'
  public void writeFooter (string experiment){..}

  // Schreibe eine beliebige Zeile mit Inhalt "'line"'
  public void writeLine (string line){..}
```

Solange keine zweite Datei geöffnet werden soll, müssen die Methoden open und close nicht weiter betrachtet werden. Diese sowie „writeHeader“ werden automatisch ausgeführt.

Beispieldatei

Im Folgenden wird ein Beispielexperiment mit einer Dateiausgabe und deren Output dargestellt:

```
public class WriteExperiment : Experiment {

  public override void initValues ()
  {
    name = "Write Experiment";
    input = "input.txt";
    output = "output.txt";
  }

  public override void init(){
    phases.Add (new ExperimentPhase (1, write));
  }

  private void write ()
  {
    writer.writeLine("Dies ist ein Test");
    writer.writeFooter(name);
  }
}
```

(Das Experiment schreibt nach dem Start eine Zeile und beendet sich nach 1 Sekunde) Folgende Ausgabedatei wird daraufhin erstellt (Platzhalter für Zeit verwendet):

```
>>>>>>>>>>>>>>> Write Experiment >>>>>>>>>>>>>>>
```

```
2 ------ Begin: M/TT/JJJJ s:mm:ss PM ------
3 Dies ist ein Test
4 ------ End: M/TT/JJJJ s:mm:ss PM ------
5 <<<<<<<<<<<<<< Write Experiment <<<<<<<<<<<<<<
```

4.5 GUI Elemente

GUI Elememente sind Skripte welche von GUIElement erben. Das Standard GUI Element welches verwendet wird ist GUIDefault. In diesem können Blackout, Instruktionstext und Statustext beeinflusst werden. Diese können zur einfachen Verwaltung durch direkte Variablen (isBlackout, status, instuction) gesetzt werden. Eine erweiterte Möglichkeit ist das verwenden des GUIDefault Objektes. Ein solches Objekt welches wie GUIDefault von GUIElement erbt, besitzt die Methoden Show() und Hide() zur Darstellung der Objekte.
Ein solches GUIElement kann wie die unter „PsyWueVR/Prefabs/“ zu findenenden Elemente „GUIDefault“ oder „GUIRate“ aufgebaut werden. Neue GUI Elemente sollten dem Experiment als public Objekte übergeben werden.