|  |
| --- |
| http://blog.gfx47.com/wp-content/uploads/2011/02/unity3d1.jpg |
| Arbeitsdokument „unityserver 3d im cave“  Hier steht ein Untertitel  **Art der Arbeit (Semesterarbeit / Bachelorthesis / etc.)** |
| Studiengang: [Studiengang einfügen]  Autor: [Autor/en einfügen]  Betreuer: [Betreuer einfügen]  Auftraggeber: [Auftraggeber einfügen]  Experten: [Experte/n einfügen]  Datum: [Datum einfügen] |

Management Summary

Inhaltsverzeichnis

[1 Einführung 4](#_Toc434763771)

[1.1 Vorarbeiten Projekt 2 4](#_Toc434763772)

[2 Infrastruktur 4](#_Toc434763773)

[2.1 Trackingsystem WorldViz 5](#_Toc434763774)

[2.2 Devices 5](#_Toc434763775)

[2.3 Unity Server 7](#_Toc434763776)

[2.4 Audio 7](#_Toc434763777)

[2.5 Video Matrix Switch 7](#_Toc434763778)

[3 Stereoskopie 8](#_Toc434763779)

[4 Immersion 8](#_Toc434763780)

[5 Devices 8](#_Toc434763781)

[6 VRPN 8](#_Toc434763782)

[7 Unity Package 8](#_Toc434763783)

[8 „Architektur der Komponenten“ 8](#_Toc434763784)

[9 Warping (optional, ev. bei Immersion) 8](#_Toc434763785)

[10 Demo Apps 9](#_Toc434763786)

[10.1 Shooting Gallery 9](#_Toc434763787)

[10.2 Model-Viewer 9](#_Toc434763788)

[10.3 App Drittpartei 9](#_Toc434763789)

[11 Abbildungsverzeichnis 9](#_Toc434763790)

[12 Tabellenverzeichnis 9](#_Toc434763791)

[13 Glossar 10](#_Toc434763792)

[14 Literaturverzeichnis 10](#_Toc434763793)

[15 Anhang 11](#_Toc434763794)

[16 Selbständigkeitserklärung 12](#_Toc434763795)

# Einführung

## Vorarbeiten Projekt 2

# Infrastruktur

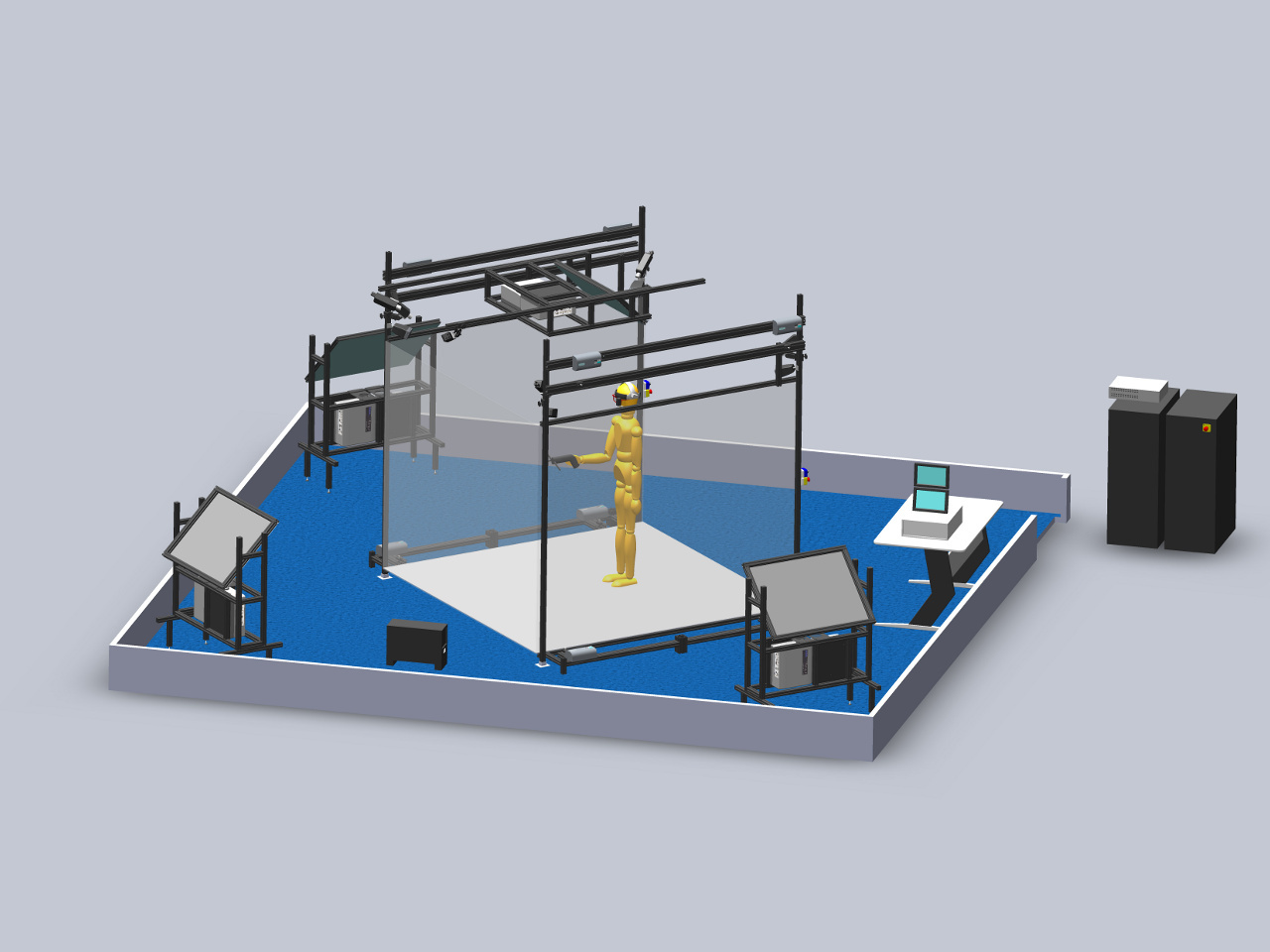


Abbildung : CAVE BFH

Der virtual reality haptic CAVE der BFH bietet Entwicklern und Forschern ein mächtiges Instrument, um hochrealistische immersive Applikationen zu bauen. Der CAVE ist eine kubische Konfiguration mit vier Leinwänden (Links, Front, Rechts, Boden) die von je 2 Projektoren bestrahlt werden. Um eine realistische 3D Projektion zu erschaffen, sind bei den Projektoren Polfilter angebracht und die Benutzer tragen eine polarisierte Brille.

Alle Komponenten vereint und dessen Abhängigkeiten werden in der folgenden Grafik dargestellt:

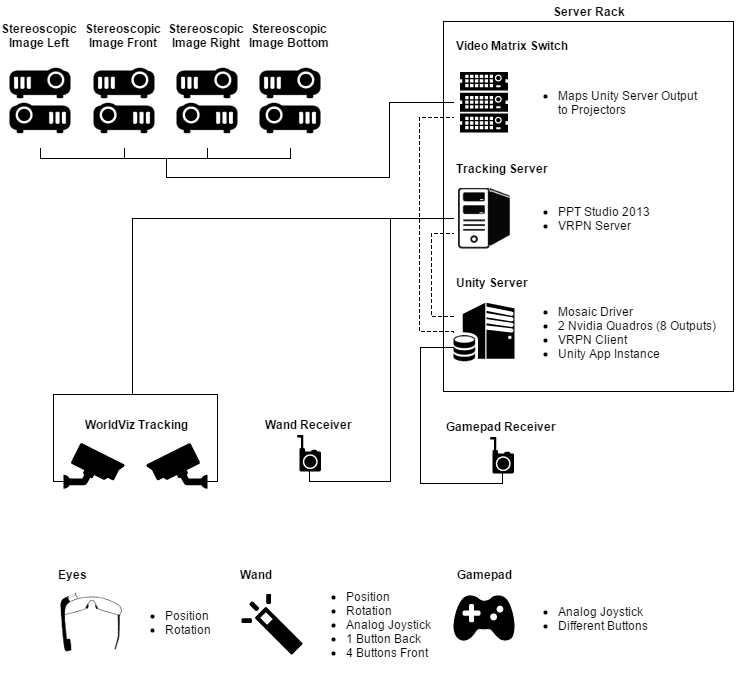


Abbildung : Infrastruktur CAVE

## Trackingsystem WorldViz

Das Trackingsystem von WorldViz wurde integriert, um Positionen und Rotationen von verschiedenen Devices im CAVE zu erfassen. Die von den 10 WorldViz Kameras übermittelten Daten werden vom PPT Studio 2013 zentral auf einem Server interpretiert, d.h. es werden Punkte im Raum und dessen Rotation der Devices berechnet, und können bei Bedarf über das VRPN-Protokoll abgefragt werden.

## Devices

* **Eyes**

Die Eyes von WorldViz sind polarisierte Brillen mit zwei montierten Infrarot-Trackern. Somit lassen sich die Position des Kopfes und dessen Rotation auf zwei Achsen (Yaw und Roll) bestimmen.

****

Abbildung : PPT Eyes, Quelle: www.worldviz.com

* **Wand**

Der Wand von WorldViz ist das primäre Eingabegerät. Nebst zwei Infrarot-Tackern, welche für die Positions- und Rotationsbestimmung verwendet werden, ist ein Gyrometer integriert, um noch präziser Drehungen feststellen zu können. Dadurch wird auch die fehlende Rotationsachse (fehlende bei den Eyes) kompensiert und es können Yaw, Roll und Pitch ermittelt werden.

Als Inputs dienen ein analoger Joystick, vier Buttons auf der Vorderseite sowie ein Button auf der Rückseite.

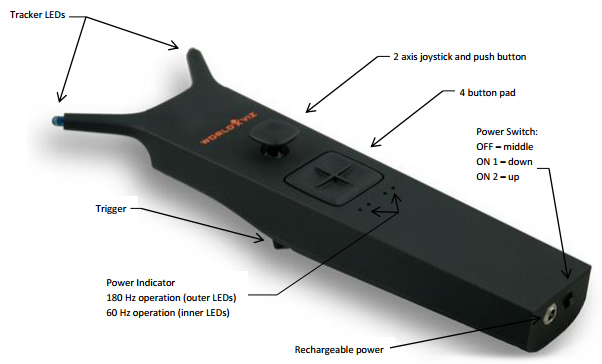


Abbildung : PPT Wand, Quelle: www.worldviz.com

* **Gamepad**

Ein weiteres Inputgerät ist ein handelsübliches Gamepad. Frei von jeglichen Tracking wird einzig die Unity-Applikation gesteuert.



Abbildung : Gamepad, Quelle: www.androidrundown.com

## Unity Server

Der leistungsstarke Untiy Server ist der Knotenpunkt des gesamten Systems. Auf diesem Rechner läuft die Unity-Applikation mit dem konfigurierten UnityPlugin, welches die Trackingdaten vom Trackingserver abgreift und das korrekte Rendering in der Uniy-Applikation für die Projektorenaufteilung sicherstellt. Mittels Mosaic, einem speziellen Treiber von Nvidia der die Aufteilung auf mehrere Grafikkartenausgänge übernimmt, werden alle Projektoren korrekt für die stereoskopische Projektion angesprochen.

## Audio

Um die Immersion weiter zu steigern, ist ein 3D Soundystem mit vier Lautsprechern in Betrieb.

## Video Matrix Switch

Weil parallel weitere Clients in Betrieb sind und Bilddaten für den CAVE liefern können, wird ein Video Matrix Switch eingesetzt, um die verschiedenen Inputquellen auf die 8 bestehenden Projektoren abbilden zu können.

# Stereoskopie

* Kamerasettings
  + Sekundäre Kamera
  + GUI Elemente
* Mosaic Settings

# Immersion

* Frustum angleichen
* Kalibrieren

# Devices

## Wand

Die Interpretation des Wands ist ein zentrales Element für die Verwendung des CAVEs mit Unity. Er dient dazu, Objekte in der virtuellen Welt zu bewegen oder zu rotieren.

* **Virtueller CAVE**

Damit die Verarbeitung der vom Wand gelieferten Informationen vereinfacht werden können und eine visuelle Darstellung möglich ist, wurde ein virtueller CAVE in Unity erstellt, welcher dieselben Dimensionen wie der reale CAVE der BFH hat. Diese genaue Adaptierung ist möglich, weil in Unity die verwendete Grösseneinheit einem Meter in der realen Welt entspricht.

Implementiert wurde das mittels einem Prefab, welches einmalig in der Hierarchie liegt.

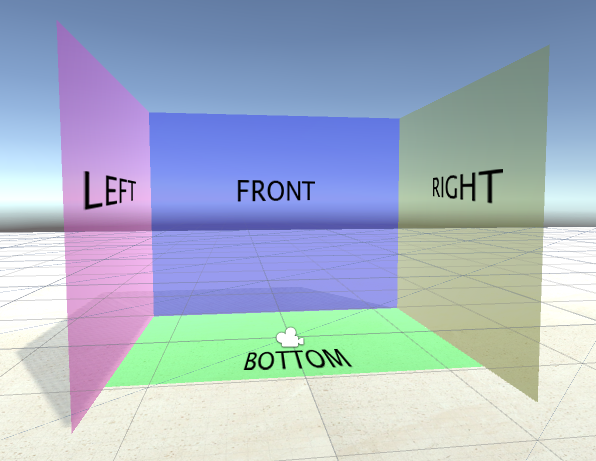


Abbildung : Unity Plugin, virtueller CAVE

* **Mousecursor**

Im virtuellen CAVE wird das Wand-Device ebenfalls virtuell abgebildet. Um möglichst generisch die Steuerung der Applikationen übernehmen zu können, ist es unerlässlich, die Mausposition mittels Wand setzen zu können, weil das häufig das primäre Inputgerät ist. Wird auf dieser Ebene des Betriebssystemes bereits Hand angelegt, entfallen spezifische Mappings auf Applikationslevel um die Steuerung übernehmen zu können.

Dazu wird ermittelt, wohin der Wand zielt. Die Verlängerung der x-Achse, also eine Gerade definiert durch die Rotation des Wands, kann einen Schnittpunkt mit einer Leinwand des CAVEs haben.

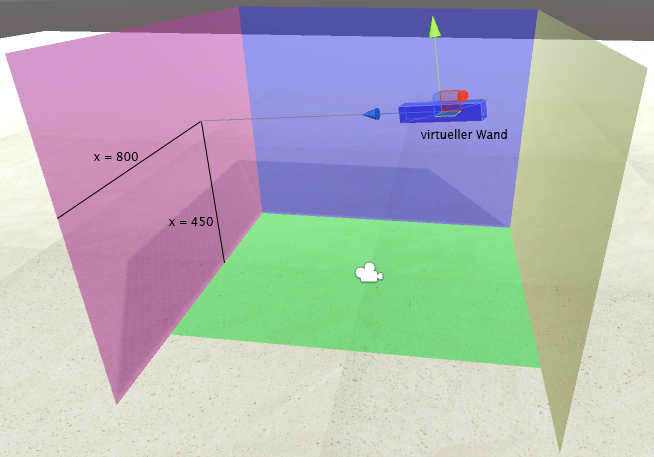


Abbildung : Unity Plugin, Schnittpunkt Wand / CAVE

Der Schnittpunkt wird von Unity mit einem Raycast ermittelt.

Ausgehend vom virtuellen Wand wird ein sogenannter Ray geschossen, welcher wahlweise nach der ersten Kollision abbricht und das getroffene Hit-Objekt zurückgibt oder durch sämtliche Colliders weiterfliegt und alle Ergebnisse als Returnwert liefert.

Der exakte Schnittpunkt auf der getroffenen Fläche liefert nun die benötigten Informationen, um den Mousecursor auf Betriebssystemebene festzulegen. Zunächst muss aber noch unterschieden werden, welche Leinwand betroffen ist. Das Mapping funktioniert so, dass bei einem Auftreffen auf die linke Leinwand der Cursor im ersten, oberen Achtel des Bildschirms festgelegt wird und der Cursor auf den zweiten, oberen Achtel dupliziert wird, damit bei der stereoskopischen Projektion beide Augen den Cursor sehen. Bei der Front-Leinwand dasselbe mit dem 3. und 4. Achtel. Findet der Schnittpunkt auf der rechten oder unteren Leinwand statt, wird der Cursor in der unteren Hälfte des Betriebssystems platziert.

*TODO Grafik Mapping CAVE-Leinwände / Bildschirm*

* Einstellung in Unity (UnityPlugin -> Inspector):  
    
  Channel = markerID (PPTStudio) – 1  
  WorldVizObject = PPT0

## Eyes

## Gamepad

# VRPN

* Wrapper

# Unity Plugin

## Konfiguration

Um eine möglichst weite Bandbreite von Unity-Applikationen abdecken zu können, werden etliche Einstellungsmöglichkeiten zur Verfügung gestellt. Diese gliedern sich in vier relevante Sektionen.

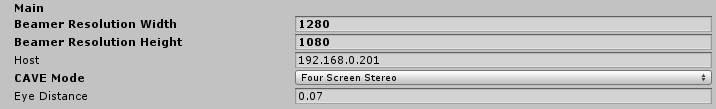
* **Allgemein**  
    
  

Abbildung 8: Unity Plugin, Settings General

Die allgemeinen Einstellungen können in den meisten Fällen unangetastet bleiben.

* **Wand**

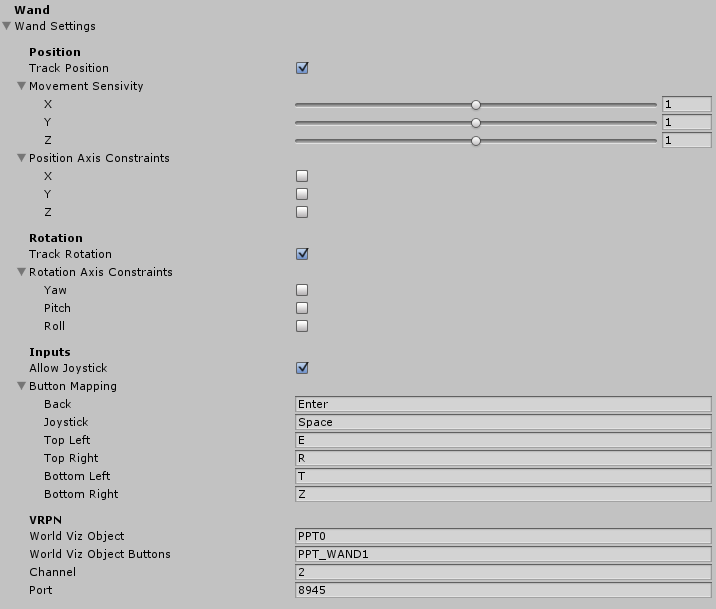


Abbildung : Unity Plugin, Settings Wand

**Position**

Falls die Position des Wands in der aktuellen Applikation unerheblich ist, kann die an dieser Stelle deaktiviert werden. Somit übernimmt der virtuelle Wand, welcher im Prefab liegt, keine Translationen vom realen Wand. Ist diese Option aber aktiviert, besteht die Möglichkeit, achsenabhängig die Sensibilität einzustellen. Das heisst, bei einer hohen Sensibilität auf der y-Achse vollführt der virtuelle Wand eine grosse Bewegung im Vergleich zu der realen Bewegung im CAVE. Gegenteilig, wird der Regler unter 0 gestellt, ist die virtuelle Bewegung kleiner als die reale Bewegung. Zusätzlich können einzelne Achsen auch komplett deaktiviert werden.

**Rotation**

Auch die Rotation kann auf Wunsch komplett oder nur selektiv deaktiviert werden.

**Inputs**

Der Wand verfügt über mehrere Buttons, die frei auf die Applikation abgebildet werden können. Zusätzlich kann der Joystick aktiviert oder deaktiviert werden.

**VRPN**

Der VRPN-Block wird für die Anmeldeinformationen beim VRPN-Server verwendet. Diese müssen nur bei einer Umstellung des PPT-Studios (auf dem Tracking-Server) adaptiert werden.

* **Eyes**

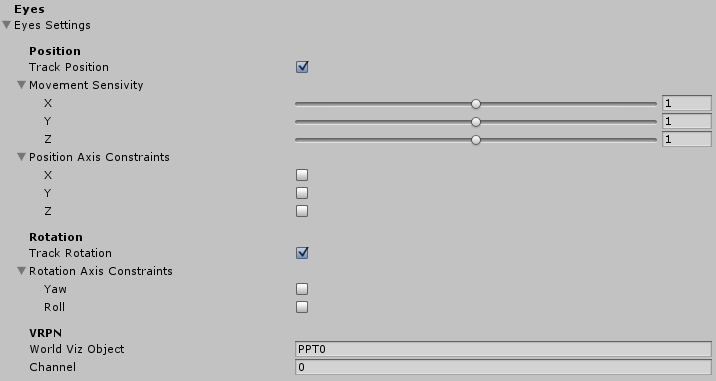


Abbildung 10: Unity Plugin, Settings Eyes

**Position**

Falls die Position der Eyes in der aktuellen Applikation unerheblich ist, kann die an dieser Stelle deaktiviert werden. Somit übernehmen die virtuellen Eyes, welche im Prefab liegen, keine Translationen von den realen Eyes. Ist diese Option aber aktiviert, besteht die Möglichkeit, achsenabhängig die Sensibilität einzustellen. Das heisst, bei einer hohen Sensibilität auf der y-Achse vollführen die virtuellen Eyes eine grosse Bewegung im Vergleich zu der realen Bewegung im CAVE. Gegenteilig, wird der Regler unter 0 gestellt, ist die virtuelle Bewegung kleiner als die reale Bewegung. Zusätzlich können einzelne Achsen auch komplett deaktiviert werden.

**Rotation**

Auch die Rotation kann auf Wunsch komplett oder nur selektiv deaktiviert werden.

**VRPN**

Der VRPN-Block wird für die Anmeldeinformationen beim VRPN-Server verwendet. Diese müssen nur bei einer Umstellung des PPT-Studios (auf dem Tracking-Server) adaptiert werden.

* **Gamepad**

*TODO add Gamepad Settings*

* **System**

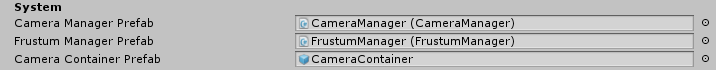


Abbildung : Unity Plugin, Settings System

Die zugewiesenen Prefabs werden beim Initialisieren des Plugins instanziiert und müssen kaum adaptiert werden.

Das CameraContainer Gameobject beinhaltet alle Kameras, die sich jeweils der Hauptkamera unterordnen und die Bildaufteilung für die verschiedenen Beamer übernimmt.

## Struktur

Der zentrale Knoten des Plugins ist die Klasse „CaveMain“. Hier sind alle Einstellungsmöglichkeiten, die Referenzen auf sämtliche Objekte und die Geometrie der virtuellen CAVEs gespeichert. *TODO Inwiefern weitere Erklärungen der Klassen? (Künzler fragen)*

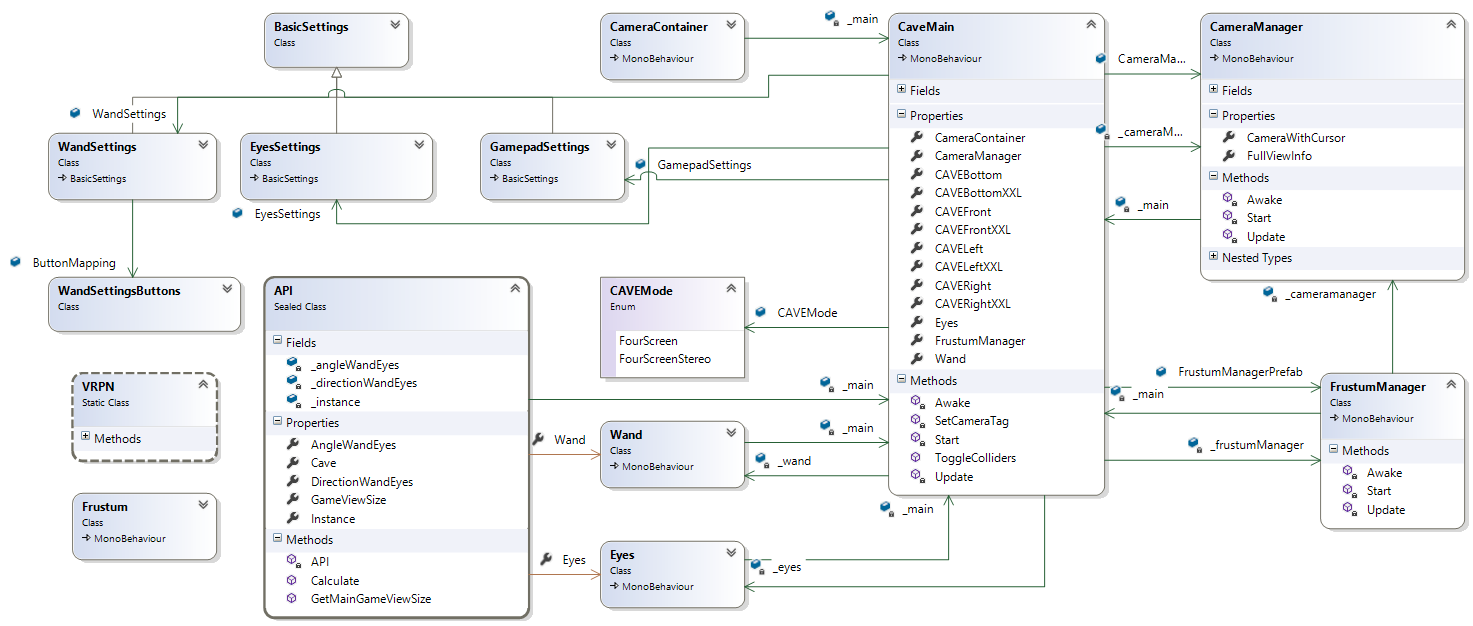


Abbildung : Unity Plugin, Klassendiagramm

## Deployment

Das entwickelte Unity Plugin muss möglichst unkompliziert und rasch in die gewünschte Applikation integriert werden können. Um das zu erreichen, wird der gleiche Ansatz wie das Deployment über den integrierten Unity Asset Store gewählt. Dazu werden sämtliche Verzeichnisse und Dateien in eine .unitypackage-Datei gepackt und können in jedes beliebige Projekt importiert werden.

Um diese Bündelung der Dateien zu erreichen, gibt es in Unity den Befehl, ein Package zu exportieren.

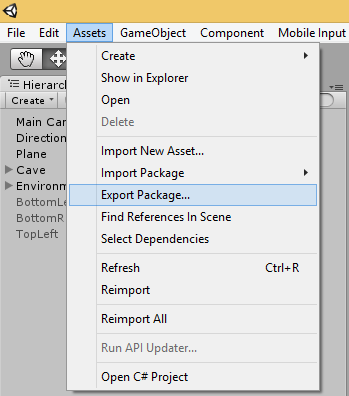


Abbildung : Unity Plugin, Export Package

Beim anschliessenden Popupmenu alle Assets auswählen und den Export starten.

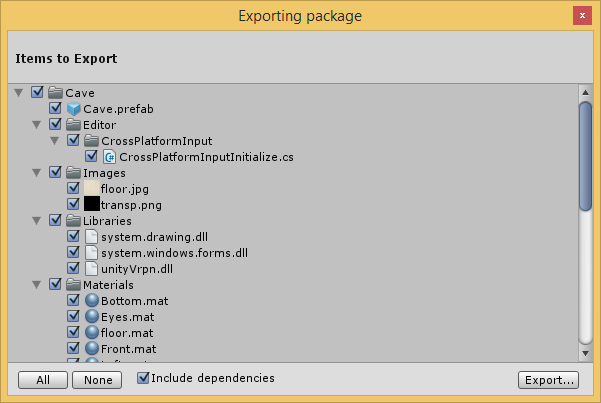


Abbildung 14: Unity Plugin, Export Package Selection

Die Verwendung des Plugins mit sämtlichen Abhängigkeiten erfolgt in wenigen Schritten:

1. **Import des Plugins**

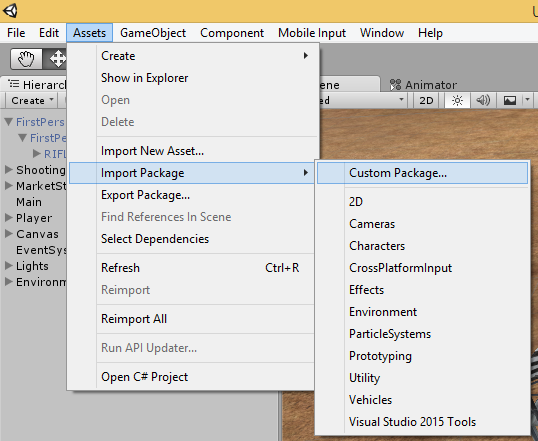


Abbildung : Unity Plugin, Import Package

Beim anschliessenden Popupmenu sind alle Assets auszuwählen und mit „Import“ zu bestätigen.

**Troubleshooting**

Abhängig der Unity-Version kann es beim Import zu Problemen führen. Sind gewisse, bereits existierende Files bereits in Verwendung, werden die Klassen / Prefabs / usw. nicht überschrieben, sondern lediglich hinzugefügt (CaveMain 1.cs, CaveMain 2.cs usw.). Um das zu umgehen, bitte das Unity-Projekt schliessen, direkt mit dem Explorer den kompletten alten Cave-Asset Folder löschen und die neue Version reinkopieren.

1. **Cave-Prefab**

In einem zweiten Schritt gilt es, das Cave-Prefab, welches sich direkt im Verzeichnis „Cave“ befindet, an einer beliebigen Stelle in der Szene zu platzieren.

1. **Einsatzbereit**

Nach diesen zwei einfachen Schritten wurde das Plugin erfolgreich in die Applikation eingebettet und kann verwendet werden.

## API

Grundsätzlich kann der gesamte Sourcecode des Plugins eingesehen und verändert werden. Um die Verwendung jedoch zu vereinfachen und die eigene Anwendung applikationsspezifisch mit dem Plugin zu verknüpfen, werden gewissen Werte, Berechnungen und Objekte in der API zur Verfügung gestellt. Mittels Singleton-Pattern wird sichergestellt, dass die Verwaltung der besagten Properties zentral an einem Ort geschieht und dort abgegriffen werden können.



Folgender Beispielcode zeigt, wie via API die Position des Wands ausgelesen wird.



# „Architektur der Komponenten“

* Unterteilung Module
  + Singleton (Hilfsklasse)
* Logisch und technisch

# Warping (optional, ev. bei Immersion)

# Demo Apps

## Shooting Gallery



Abbildung : Shooting Gallery Ingame

Das Setting dieses Demospiels ist eine Schiessbude im Wilder Westen Stil, wie sie auf einem Jahrmarkt anzutreffen ist. Die Galerien mit den abzuschiessenden Zielen verteilen sich rund um den Spieler. Mit Hilfe des Head Trackings kann sich der Spieler in der gesamten Szenerie umschauen, Bewegungen ausführen und die Objekte aus verschiedenen Perspektiven betrachten. Das Wand-Device steuert das Gewehr, um die Zielobjekte anzuvisieren und abzuschiessen. Die Buttons des Wands werden gebraucht um das Gewehr abzufeuern.

* **Umgebung**Die Landschaft ist ein Model aus dem Asset Store von Unity mit riesigen Ausmessungen. Um die abzuschiessenden Ziele platzieren zu können, Hindernisse zu schaffen und der Szenerie Leben einzuhauchen, wurden verschiedene Marktstände, Heukarren, Büsche und Zäune eingefügt.

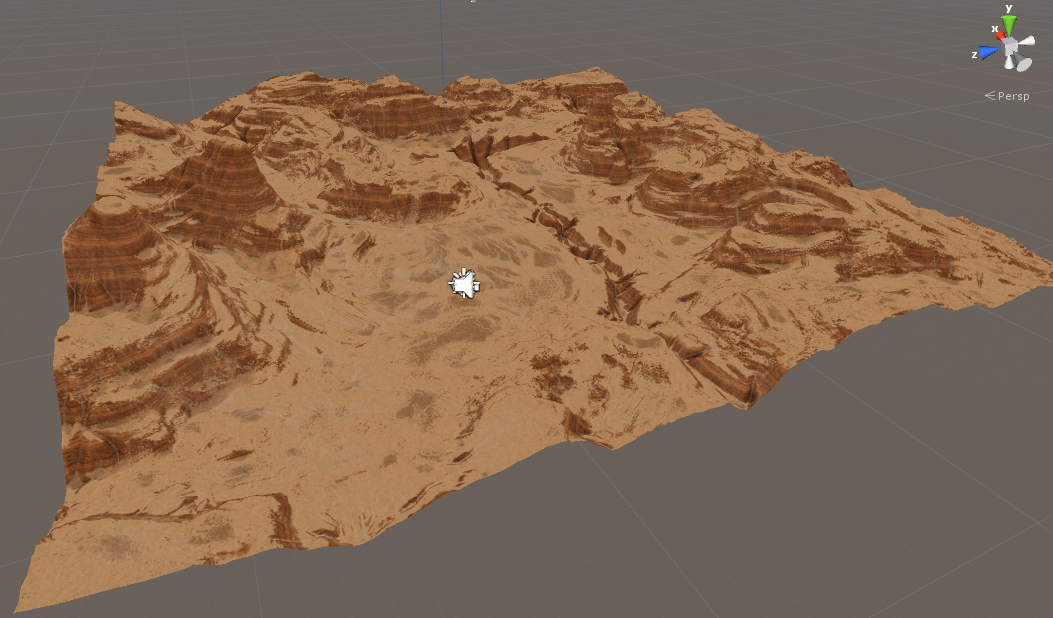


Abbildung : Desert Model, Quelle: Unity Asset Store

* **Gewehr**

Das Model des Gewehrs ist ebenfalls aus dem Unity Asset Store. Die Enten und Zielscheiben werden mit der Cursorposition anvisiert, welche durch das UnityPlugin vom Wand gesetzt wird.

**Rotation**

Das Gewehr dreht sich entsprechend dessen Position. Die Rotation wird zweierlei beeinflusst.

1. **Rotation der Eyes**

Basierend auf der Rotation der Eyes auf der y-Achse (Yaw) und der z-Achse (Roll) dreht sich auch das Gewehr im Spiel. Somit wird ermöglicht, dass sich der Benutzer des CAVEs drehen kann und das Gewehr immer in seine Blickrichtung zielt. Neigt er den Kopf leicht auf eine Seite, übernimmt das Gewehr ebenfalls diese Manipulation rollt sich auf die Seite.

1. **Relativer Winkel zwischen Eyes und Wand**

Zusätzlich zu der Blickrichtung, welche mittels Eyes festgestellt wird, folgt das Gewehr der aktuellen Cursorposition. Dazu wird der relative Winkel zwischen dem Wand und den Eyes berechnet und darauf basierend erfolgt eine zusätzliche Rotation des Gewehrs.

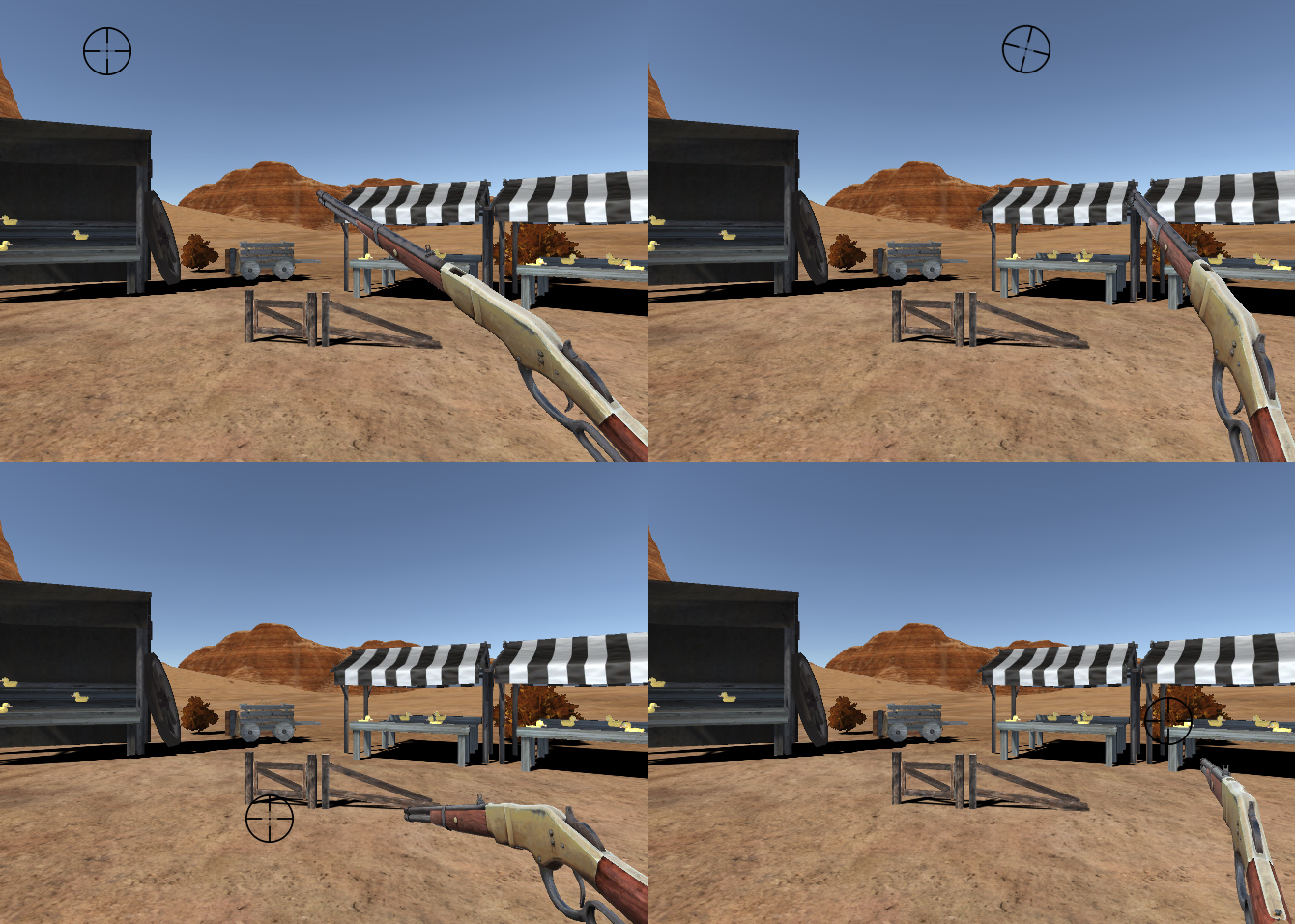


Abbildung : Shooting Gallery, Rotation des Gewehrs

**Schuss**

Dort wo sich der Cursor auf der 2D-Ebene befindet, wird auch präzise der Schuss in der 3D-Welt auftreffen. Dazu wird ein Raycast mit der Ausrichtung der Kamera an der Position des Cursors in die Umgebung geschossen und geschaut, welches Objekt als erstes im Wege steht.

Beim Auftreffen dieses virtuellen Schusses wird dem Ziel mitgeteilt, ob nun eine Interaktion erfolgen soll oder nicht. Zusätzlich wird ein Partikeleffekt, welcher den Einschuss verdeutlich, an der getroffenen Stelle erzeugt. Ein weiterer Raucheffekt wird bei der Flinte direkt gezeigt.



Abbildung : Shooting Gallery, Rauch

**Audio**

* **Zielscheibe**

Eines der beiden abzuschiessenden Ziele ist eine Holzkonstruktion mit drei Zielscheiben, welche sich zu zufälligen Zeitpunkten nach oben, bzw. nach unten klappen und somit angreifbar, bzw. nicht angreifbar werden.

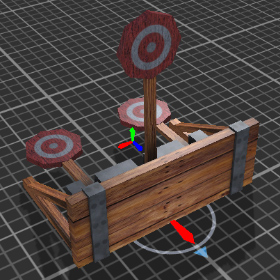


Abbildung : Shooting Gallery, Zielscheibe

Die Animationen sind im FBX-Model gespeichert und werden mittels einem AnimationController ausgelöst. Zu zufälligen Zeitpunkten wird eine der Show-States aktiviert um die Animation abzuspielen. Trifft während einer gewissen Zeitspanne kein Schuss die Zielscheibe, aktiviert sich der Hide-State und geht anschliessend zurück in den Idle-State. Im Gegenzug, trifft der Spieler auf die Zielscheibe, wird die Animation beim Hit-State abgespielt und es werden Punkte gutgeschrieben. Zudem wird als Audiofeedback ein entsprechender Sound gehört. Anschliessend fängt die Sequenz von vorne an.

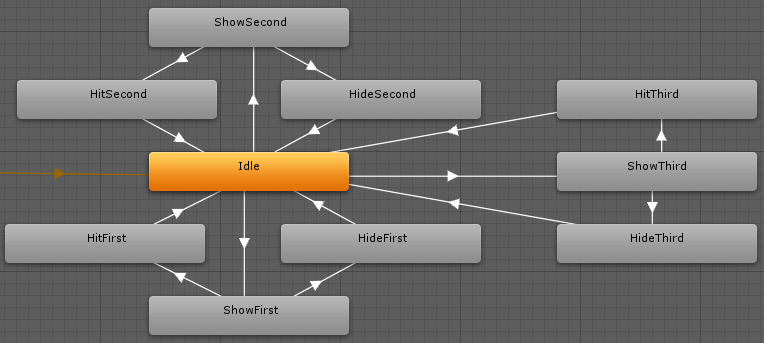


Abbildung : Shooting Gallery, Zielscheibe Animation Controller

* **Ente**

Das zweite abzuschiessende Objekt ist eine Gummiente, die sich auf einer festgelegten Bahn mit zufälliger Geschwindigkeit nach vorne und hinten bewegt.



Abbildung : Shooting Gallery, Ente

Am Anfang und Ende der Bahn befinden sich jeweils Trigger, die ausgelöst werden, sobald sich das Mesh der Ente damit überschneidet. In diesem Moment ändert sich die Bewegungsrichtung der Ente und eine neue Geschwindigkeit wird zufällig zwischen einem definierten Bereich gewählt.

Bei einem Treffer wird der Winkel zwischen der aktuellen Kamera und der Ente berechnet und entsprechend ein Impuls auf die Ente angewandt, damit sie in die korrekte Richtung davonfliegt. Sobald die Ente keinen Kontakt mit der Oberfläche mehr hat auf der sie sich bewegt, wird ein Event losgeschickt um nach einer gewissen Zeitspanne die Position und Rotation wiederherzustellen und Punkte gutzuschreiben. Bei einem Treffen ist zudem ein Audiofeedback (Quaken) zu hören.

Weil das gefundene Model der Ente für die Anwendungszwecke viel zu detailliert war, musste aus Performancegründen mit Blender eine Reduktion der Faces erfolgen. Von ehemals 7160 sind noch 1288 Faces übrig geblieben. Dank der Textur und der Distanz, die zwischen dem Spieler und der Ente liegt, fällt der Unterschied nicht auf.

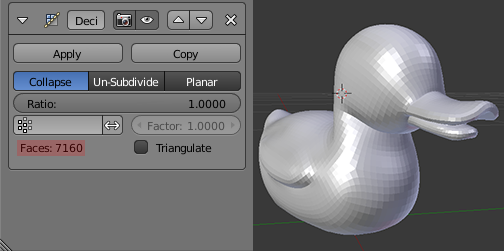


Abbildung : Shooting Gallery, Ente mit vielen Details

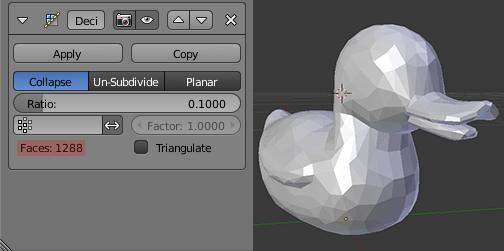


Abbildung : Shooting Gallery, Ente mit reduzierten Details

* **Punktesystem**

Nebst der bereits verstrichenen Zeit ist in einer Ecke als UI-Element die erspielte Punktzahl sichtbar. Die beiden abzuschiessenden Ziele (Ente und Zielscheibe) stellen öffentliche, statische Delegates (Events) zur Verfügung, mittels denen andere Objekte Methoden registrieren können. Wird also beispielsweise eine beliebige Ente abgeschossen, wird die Methode aufgerufen.

Die Klasse, welche für die Punkteberechnung und –darstellung zuständig ist, registriert entsprechend eine Methode, welche als Parameter die erspielten Punkte erhält. Somit kann dort sauber Buch geführt werden über den aktuellen Punktestand und ihn als GUI-Element anzeigen.



## Model-Viewer

## App Drittpartei

# Abbildungsverzeichnis

[Abbildung 1: CAVE BFH 4](#_Toc435960549)

[Abbildung 2: Infrastruktur CAVE 5](#_Toc435960550)

[Abbildung 3: PPT Eyes, Quelle: www.worldviz.com 6](#_Toc435960551)

[Abbildung 4: PPT Wand, Quelle: www.worldviz.com 6](#_Toc435960552)

[Abbildung 5: Gamepad, Quelle: www.androidrundown.com 7](#_Toc435960553)

[Abbildung 6: Unity Plugin, Settings General 11](#_Toc435960554)

[Abbildung 7: Unity Plugin, Settings Wand 11](#_Toc435960555)

[Abbildung 8: Unity Plugin, Settings Eyes 12](#_Toc435960556)

[Abbildung 9: Unity Plugin, Settings System 13](#_Toc435960557)

[Abbildung 10: Unity Plugin, Klassendiagramm 14](#_Toc435960558)

[Abbildung 11: Unity Plugin, Export Package 15](#_Toc435960559)

[Abbildung 12: Unity Plugin, Export Package Selection 16](#_Toc435960560)

[Abbildung 13: Unity Plugin, Import Package 17](#_Toc435960561)

[Abbildung 14: Shooting Gallery Ingame 20](#_Toc435960562)

[Abbildung 15: Desert Model, Quelle: Unity Asset Store 21](#_Toc435960563)

[Abbildung 16: Shooting Gallery, Rotation des Gewehrs 22](#_Toc435960564)

[Abbildung 17: Shooting Gallery, Rauch 22](#_Toc435960565)

[Abbildung 18: Shooting Gallery, Zielscheibe 23](#_Toc435960566)

[Abbildung 19: Shooting Gallery, Zielscheibe Animation Controller 23](#_Toc435960567)

[Abbildung 20: Shooting Gallery, Ente 24](#_Toc435960568)

[Abbildung 21: Shooting Gallery, Ente mit vielen Details 24](#_Toc435960569)

[Abbildung 22: Shooting Gallery, Ente mit reduzierten Details 25](#_Toc435960570)

# Tabellenverzeichnis

[Tabelle 1: Et ut aut isti repuditis qui ium 3](#_Toc371572893)

# Glossar

**Auinweon**

Et ut aut isti repuditis qui ium 7

**Batnwpe**

Et ut aut isti repuditis qui ium 9

**Cowoll**

Et ut aut isti repuditis qui ium 11

# Literaturverzeichnis

**Literatureintrag**

*Autorname, Autorvorname, Buchtitel, Verlag, Ort, Ausgabe, Jahr* 7

**Literatureintrag**

*Autorname, Autorvorname, Buchtitel, Verlag, Ort, Ausgabe, Jahr* 9

**Literatureintrag**

*Autorname, Autorvorname, Buchtitel, Verlag, Ort, Ausgabe, Jahr* 11

# Anhang

Et ut aut isti repuditis qui ium nonsecturia quis incientiae laborem elliquis et quatur, sitiur aut od moluptatur aut ea conseque peri sim erro essequisit remporia dem et landi dest, cone poris quunt volecab ipidero quatur ad quibusamus.

# Selbständigkeitserklärung

Ich bestätige, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der im Literaturverzeichnis angegebenen Quellen und Hilfsmittel angefertigt habe. Sämtliche Textstellen, die nicht von mir stammen, sind als Zitate gekennzeichnet und mit dem genauen Hinweis auf ihre Herkunft versehen.

Ort, Datum:

Unterschrift: