Systemspezifikation Vortex-Tunnel

|  |  |  |
| --- | --- | --- |
| **Version** | **Beschreibung** | **Bearbeiter** |
| 1.0 | Erster Entwurf des Berichts | Marc Nussbaumer |
|  |  |  |
|  |  |  |
|  |  |  |

Inhaltsverzeichnis

[Einführung 4](#_Toc501706687)

[Anforderungen 4](#_Toc501706688)

[Scope 4](#_Toc501706689)

[Use Cases 6](#_Toc501706690)

[Use Case Diagramm 6](#_Toc501706691)

[Use Case: Konfigurieren 6](#_Toc501706692)

[Use Case: Tunnel durchlaufen 6](#_Toc501706693)

[Allgemeine Entwurfsentscheide 8](#_Toc501706694)

[Effektgenerierung der Drehung 8](#_Toc501706695)

[Prototyp Pointlight 8](#_Toc501706696)

[Prototyp Spotlight 8](#_Toc501706697)

[Prototyp Textur 8](#_Toc501706698)

[Varianten zur Parameter-Eingabe 8](#_Toc501706699)

[Variante 1: Eingabemaske beim Start der Tunnel-Simulation 8](#_Toc501706700)

[Variante 2: Parameter in einem XML-File 8](#_Toc501706701)

[Variante 3: Ein simpler Editor zum Tunnelbau 9](#_Toc501706702)

[Vergleich der drei Varianten 9](#_Toc501706703)

[Konfigurierbarkeit Steg 10](#_Toc501706704)

[Wahl der Tunneltextur 10](#_Toc501706705)

[Tunnelgenerierung 11](#_Toc501706706)

[Systemübersicht 13](#_Toc501706707)

[Komponentendiagramm 13](#_Toc501706708)

[Hardware-Anforderungen 13](#_Toc501706709)

[Schnittstellen 14](#_Toc501706710)

[Konfigurations-Datei 15](#_Toc501706711)

[Struktur 15](#_Toc501706712)

[Vortex-Tunnel 16](#_Toc501706713)

[Abschnitte 16](#_Toc501706714)

[Abschnitt 17](#_Toc501706715)

[Wandmuster 17](#_Toc501706716)

[Textur 18](#_Toc501706717)

[Lichter 19](#_Toc501706718)

[Farbe 20](#_Toc501706719)

[Stegtransparenz 21](#_Toc501706720)

[Rotation der Lichter 22](#_Toc501706721)

[Drehende Textur 23](#_Toc501706722)

[Umgang mit Renderer 23](#_Toc501706723)

[UV Mapping 24](#_Toc501706724)

[SketchUp Modell zu Unity 25](#_Toc501706725)

[Transformation Elemente 26](#_Toc501706726)

[XML-Mapping 28](#_Toc501706727)

[Best Practices VR 29](#_Toc501706728)

[Deployment 30](#_Toc501706729)

[Ausblick 31](#_Toc501706730)

# Einführung

Im Auftrag des iHomeLab der Hochschule Luzern soll nicht technikaffinen Personen «Virtual Reality» anhand eines sogenannten Vortex-Tunnels nähergebracht werden. Ein Vortex-Tunnel ist eine drehende Röhre, welche von einer Person auf einem Steg durchschritten werden soll. Dabei ist es in der Röhre dunkel, sichtbar sind nur sich drehende Lichtpunkte und Muster.

Mit diesem Aufbau lässt sich der Gleichgewichtssinn der Testpersonen täuschen.

## Anforderungen

Gemäss Aufgabenstellung sind folgende Ergebnisse zu erarbeiten:

* Konzept zur Erreichung des maximalen Effekts auf den Gleichgewichtssinn der Probenden
* Erstellung Softwarekonzept und Testkonzept
* Softwareprototyp mit Unity3d und Oculus Rift oder HTC Vive
* Testresultate mit den Probanden und Diskussion der erarbeiteten Ergebnisse
* Code, Dokumentation und HowTo’s auf GitHub.

Nach einem Workshop mit Herr Biallas (siehe Anhang «20171011\_Protokoll Sitzung Vortex-Tunnel») wurden folgende konkreten Anforderungen für den Tunnel definiert:

Muss:

* Benutzer soll sich durch laufen durch den Tunnel bewegen können
* Es existiert ein gerades Tunnelmodell durch welches gelaufen werden kann
* Der Tunnel soll per XML konfigurierbar sein

Kann:

* Der Tunnel soll Kurven aufweisen$

//Änderugen durch Sitzungen

## Scope

* Kurven sollen nur im 90° Winkel konzipiert werden
* Lichtmuster sollen zufällig sein
* Form der Lichtpunkte ist beliebig
* Bodenbeschaffenheit sollen durch 3 Verschiedene Konfigurationen gewählt werden
* Tunnel ist immer ein Kreis / Zylinder
* Minimale Toleranz zu Konfiguration ist mindestens 10cm

Was ist nicht Teil der Ziele (noscope)?

Änderungen während des Projektes:

Basierend auf den Erkenntnissen aus der Versuchsdurchführung (siehe Anhang «XYXXXX») wurde an der Sitzung vom 02.12.2017 folgende Änderungen an der Zielsetzung durchgeführt:

Die Kann-Anforderung der Kurve fällt weg, stattdessen soll die Konfigurierbarkeit des Stegs mit folgenden zusätzlichen Parametern erweitert werden:

* + ***Handgeländer ein/ausschaltbar***
  + ***Breite des Stegs***
  + ***Transparenz-Grad des Stegs***
  + ***Textur***
  + ***Höhe***
* -Kurve fällt weg -> Fokus auf Steg
* -Ergänzung Steg -> Versuchsresultate
* -90° Kurve und Gerader Steg -> Sitzung Klärung Aufgabenstellung
* Flexible Parameter -> Sitzung Klärung Aufgabenstellung
* Mehrere Abschnitte -> Sitzung Zwischenpräsentation

## Use Cases

### Use Case Diagramm

### Use Case: Konfigurieren

### Use Case: Tunnel durchlaufen

System

Exe zu SteamVR

Umgebung

# Allgemeine Entwurfsentscheide

## Effektgenerierung der Drehung

Als Entscheidungsbasis für das weitere Vorgehen zur Effekterzeugung wurden drei Prototypen mit unterschiedlichen Ansätzen erstellt und anschliessend in einer Sitzung vorgestellt.

### Prototyp Pointlight

PG -> Insbesondere Drehung um Mittelpunkt genauer beschreiben

### Prototyp Spotlight

PG

### Prototyp Textur

Nebst der Verwendung von vorgefertigten Unity-Lichteffekten wurde zusätzlich eine mögliche Lösung mittels Innen-Textur des Tunnel-Modells untersucht. Dabei wird auf jegliche Beleuchtung ausser dem Ambient-Light verzichtet. Stattdessen werden die Lichtpunkte durch Punkte auf der Grafik für die Textur dargestellt.

Um eine Textur auf einem Modell abzubilden ist eine sogenannte UV-Map notwendig, diese bestimmt welche Bildpunkte auf welche Modell-Punkte abgebildet werden sollen (siehe XXX).

Um die Textur zu drehen wurden folgende Wege untersucht:

* Drehung des Modells
* Drehung der Kamera
* Drehung der Textur

Der Effekt konnte durch eine Rotation an Z-Achse für die Drehung des Modells und der Kamera erzielt werden. Jedoch ist der Nachteil, dass diese Methoden nicht verwendet werden können in Zusammenhang mit Kurven. (Siehe Abbildung XYZ)

Stattdessen wurde der Ansatz in welchem die Textur selbst bewegt wird weiterverfolgt, da dort das Problem einer Kurve gelöst werden kann. (Mittels UV? Wie gelöst -> genauer erläutern)

Die zwei funktionierenden Prototypen wurden an der Sitzung vom 08.11.2017 (siehe Anhang «20171108\_Protokoll Zwischenpräsentation Vortex-Tunnel») präsentiert. Es wurde dabei entschieden beide Lösungsansätze zu verknüpfen und in zukünftigen Versionen zu verwenden.

## Varianten zur Parameter-Eingabe

In den folgenden Abschnitten werden verschiedene Varianten zur Parameter-Eingabe angeschaut und im Anschluss verglichen. Die Parameter-Eingabe muss nicht durch den Endbenutzer erfolgen, sondern durch Vertreter des iHome-Labs. Aus diesem Grund muss die Eingabe nicht zwingend über ein im Programm integriertes GUI erfolgen. Das Verändern der Parameter soll aber das Öffnen von Unity/einer IDE nicht voraussetzen.

### Variante 1: Eingabemaske beim Start der Tunnel-Simulation

Die Parameter können direkt vom Benutzer, der die VR-Brille trägt, vor Start der Simulation gesetzt werden. Unity bietet einfache Steuerelemente an, die dies ermöglichen.

### Variante 2: Parameter in einem XML-File

Die Parameter werden strukturiert in einem XML-File eingegeben. Anschliessend wird die VR-Simulation gestartet und die Werte werden aus der Datei übernommen.

### Variante 3: Ein simpler Editor zum Tunnelbau

Ein simples Tool zum Zeichnen des Tunnels. Die Benutzeroberfläche des Tools könnte den Tunnel aus der Vogelperspektive darstellen und das Setzen von neuen Tunnelabschnitten ermöglichen. Das Tool wird eine Datei erzeugen (xml, txt oder anderes Format), die dann beim Programmstart eingelesen und zur Generierung des Tunnels verwendet wird. Im Vergleich zu den ersten beiden Varianten wird bei dieser Lösung kein zufälliger Tunnel generiert.

### Vergleich der drei Varianten

Zur Auswahl der besten Variante werden alle Konzepte verglichen. Die Kriterien an eine Parametereingabe sind die folgenden:

* Einfachheit der Eingabe: Die Parametereingabe soll möglichst kein technisches Know-How erfordern. (1= Gelernter Informatiker, 3=Technischer Assistent, 5= Microsoft-Office Anwender)
* Komfort: Wie angenehm ist es für den Benutzer, die Parameter einzugeben. (1=Mehrere Zusatzprogramme notwendig, 3=Ein Zusatzprogramm notwendig, 5=Alles in der Applikation)
* Entwicklungsaufwand: Wie aufwändig ist die Entwicklung? (1= sehr aufwändig und grosser Zeitaufwand, 3=tragbarer Aufwand ohne zusätzliche Kenntnisse/Libraries, 5=wenig Aufwand und kann gut in C# umgesetzt werden.)
* Genauigkeit: Wie gezielt und präzise können die Parameter gesetzt werden? (1=grobe Definition ohne Massangabe, 3=Masse in bestimmten Intervallen, 5=freie Massangaben)
* Erweiterbarkeit: Wie einfach können neue Parameter hinzugefügt werden? (1=Entwickler für die Anpassung benötigt, 3=Technischer Assistent, 1=Laie)

Die drei Varianten wurden an den genannten Kriterien gemessen und verglichen. Jedes Kriterium wurde zusätzlich gewichtet.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Gewichtung** | **Variante 1 -GUI** | **Variante 2 -XML** | **Variante 3 -Editor** |
| **Einfachheit** | 3 | 5 | 3 | 5 |
| **Komfort** | 2 | 3 | 3 | 4 |
| **Entwicklungsaufwand** | 3 | 2 | 5 | 1 |
| **Genauigkeit** | 1 | 3 | 5 | 3 |
| **Erweiterbarkeit** | 1 | 1 | 5 | 1 |
| **Gesamt** | 10 | 31 | 40 | 30 |

Tabelle 1 Vergleich Konfigurationsmethoden

Aus dem Vergleich geht hervor, dass sich die Parameter-Eingabe über eine XML-Datei am besten eignet. Deshalb wird dieser Ansatz weiterverfolgt.

## Konfigurierbarkeit Steg

Gittersteg -> Durchsichtig vor Versuch -> Rückmeldung Zwischenpräsentation

Stegkonfiguration -> Aufgrund Rückmeldungen des Versuches (siehe Wissenschaftlicher Bericht)

## Wahl der Tunneltextur

Zuerst fix

Dann frei wählbar -> Sitzung XY

Dann Spirale -> Rückmeldungen Versuch

## Tunnelgenerierung

Zuerst Vertex

Dann aufgrund erster Sitzung -> Nur 90° Kurven (siehe Kapitel Anforderungen) -> Modell ist einfacher -> weniger Fehleranfällig

Mehrere Abschnittskonfigurationen -> Aufgrund Zwischenpräsentation

Tunnelgenerierung -> Mesh setzten wieso entscheidung für SketchUp-Modell

# Systemübersicht

## Komponentendiagramm

MN->

XML->Vortex->SteamVR -> HTC Vive

## Hardware-Anforderungen

Grafikkarte etc

# Schnittstellen

Steam-VR

EVTL Einbindung ins Projekt`?

# Konfigurations-Datei

## Struktur

<VortexTunnel durchmesser=**"6"**>

<Abschnitte>

<!--Tunnelabschnitt -->

<Abschnitt>

<Typ>**gerade**</Typ>

<Steg>**gitter.png**</Steg>

<Laenge>**20**</Laenge>

<Wandmuster>

<Textur>

<Name>**spotlight\_texture.png**</Name>

<Drehrichtung>**rechts**</Drehrichtung>

<Drehgeschwindigkeit>**4**</Drehgeschwindigkeit>

</Textur>

<Lichter>

<Reichweite>**2**</Reichweite>

<Intensitaet>**10**</Intensitaet>

<Anzahl>**50**</Anzahl>

<Drehrichtung>**rechts**</Drehrichtung>

<Drehgeschwindigkeit>**4**</Drehgeschwindigkeit>

<!--Festlegen des Farbspektrums, in welchem das Muster erzeugt wird (Werte 0-1) -->

<Farbe>

<MinimumRot>**0**</MinimumRot>

<MaximumRot>**1**</MaximumRot>

<MinimumGruen>**0**</MinimumGruen>

<MaximumGruen>**1**</MaximumGruen>

<MinimumBlau>**0**</MinimumBlau>

<MaximumBlau>**1**</MaximumBlau>

</Farbe>

</Lichter>

</Wandmuster>

</Abschnitt>

<!-- Weitere Abschnitte können hinzugefügt werden(Reihenfolge!)

<Abschnitt>

...

</Abschnitt>

-->

</Abschnitte>

</VortexTunnel>

## Vortex-Tunnel

<VortexTunnel durchmesser=**"6"**>

Attribut «**durchmesser**» legt den Durchmesser in **Meter** des gesamten Tunnels fest.

**Erlaubte Werte:**

Gleitkomma-Zahlen welche grösser als die Grösse des Tunnelbenutzers sind. Ansonsten liegt die Kamera ausserhalb des Tunnels und sieht nichts.

## Abschnitte

<Abschnitte>

<!--Tunnelabschnitt -->

<Abschnitt>

**<<Abschnitt 1>>**

</Abschnitt>

<Abschnitt>

**<<Abschnitt 2>>**

</Abschnitt>

</Abschnitte>

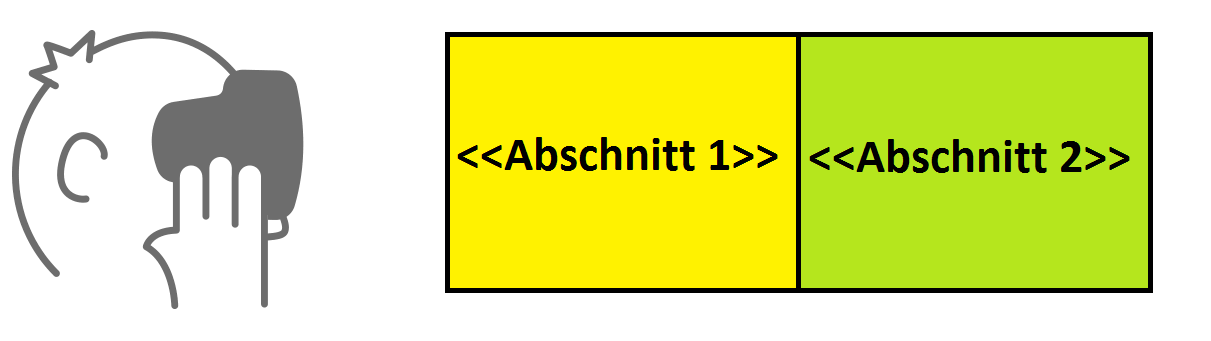
In «**Abschnitte**» werden die einzelnen Tunnel-Abschnitte (siehe weiter unten) spezifiziert. Dabei werden die Tunnel-Abschnitte von oben nach unten abgearbeitet.   


Bild-Quelle: <http://www.view-master.com/en-US/Images/troubleshooting-reel_tcm1147-254727.png>

## Abschnitt

<Abschnitt>

<Steg>**gitter.png**</Steg>

<Laenge>**20**</Laenge>

<Wandmuster>

**<<Definition Wandmuster>>**

</Wandmuster>

</Abschnitt>

«**Steg**» legt fest welche Bild-Datei als Textur für den Steg verwendet werden soll.

Erlaubte Werte:

PNG-Datei - Die PNG Datei muss folgende Eigenschaften für eine gute Darstellung

* Auflösung soll einem Vielfachen von 2 entsprechen. 40x40px ist beispielsweise okay, 41x42px jedoch nicht.
* Es muss eine PNG-Datei sein, andere Bildformate werden nicht unterstützt
* Die Datei kann durch den Namen spezifiziert werden, falls sie bei der EXE-Datei liegt. Ansonsten soll der vollständige Pfad verwendet werden.

«**Laenge**» legt fest, wie lange der Tunnel-Abschnitt in Meter sein soll.

**Erlaubte Werte:**  
  
positiver Integer - positiver Ganz-Zahlwert von 1 bis 2147483647

## Wandmuster

<Wandmuster>

<Textur>

<<Textur-Definition>>

</Textur>

<Lichter>

<<Lichter-Definition>>

</Lichter>

</Wandmuster>

Im Wandmuster werden die Eigenschaften der Tunnelwand beschrieben, dabei müssen sowohl Textur, als auch Lichter definiert sein. Ansonsten dient dieses Tag der Übersicht im XML als Trennung zwischen dem Tunnelaufbau und der Tunneldarstellung.

## Textur

<Textur>

<Name>**spotlight\_texture.png**</Name>

<Drehrichtung>**rechts**</Drehrichtung>

<Drehgeschwindigkeit>**4**</Drehgeschwindigkeit>

</Textur>

«**Name**» legt fest welche Bild-Datei als Tunneltextur für den momentanen Abschnitt verwendet werden soll.

Erlaubte Werte:

PNG-Datei - Die PNG Datei muss folgende Eigenschaften für eine gute Darstellung

* Auflösung soll einem Vielfachen von 2 entsprechen. 40x40px ist beispielsweise okay, 41x42px jedoch nicht.
* Es muss eine PNG-Datei sein, andere Bildformate werden nicht unterstützt
* Die Linke-Seite sollte direkt an die rechte Seite anschliessen können. Ansonsten gibt es Linien im Tunnel
* Die Datei kann durch den Namen spezifiziert werden, falls sie bei der EXE-Datei liegt. Ansonsten soll der vollständige Pfad verwendet werden.

«**Drehrichtung**» legt fest in welche Richtung vom Betrachter aus die Textur dreht.

**Erlaubte Werte:**

«rechts» - dreht die Textur vom Betrachter aus im Uhrzeigersinn  
«link» - dreht die Textur vom Betrachter aus im Gegenuhrzeigersinn

«**Drehgeschwindigkeit**» legt die Kreisbewegung in Unity-Meter / Sekunde fest. Zu beachten ist, dass die Textur viermal im Tunnel angezeigt wird

**Erlaubte Werte:**

Gleitkomma-Zahlen - Empfohlen sind Werte zwischen 0 und 1.

## Lichter

<Lichter>

<Reichweite>**2**</Reichweite>

<Intensitaet>**10**</Intensitaet>

<Anzahl>**50**</Anzahl>

<Drehrichtung>**rechts**</Drehrichtung><Drehgeschwindigkeit>**4**</Drehgeschwindigkeit><Farbe>

**<<Erlaubter Farbraum>>**

</Farbe>

</Lichter>

«**Reichweite**» legt fest wie viele Meter das Licht auf anderen Objekten sichtbar ist.

Erlaubte Werte:

Gleitkommazahlen - Licht ist nur bei positiven Werten sichtbar

«**Intensitaet**» legt fest wie stark eine Lichtquelle sein soll.

Erlaubte Werte:

Positiver Integer - Licht wird bei höheren Werten stärker

«**Anzahl**» legt fest wie viele Lichter in diesem Tunnelabschnitt dargestellt werden sollen.

Erlaubte Werte:

Positiver Integer - Anzahl Lichter die zufällig verteilt werden im Tunnelabschnitt

«Drehrichtung» legt fest in welche Richtung die Lichter im Tunnel-Abschnitt drehen sollen.

Erlaubte Werte:

«rechts» - dreht die Textur vom Betrachter aus im Uhrzeigersinn  
«link» - dreht die Textur vom Betrachter aus im Gegenuhrzeigersinn

«**Drehgeschwindigkeit**» legt die Drehgeschwindigkeit in Unity-Meter pro Sekunde fest

**Erlaubte Werte:**

Gleitkomma-Zahlen - Empfohlen sind Werte zwischen 0 und 1

## Farbe

<Farbe>

<MinimumRot>**0**</MinimumRot>

<MaximumRot>**1**</MaximumRot>

<MinimumGruen>**0**</MinimumGruen>

<MaximumGruen>**1**</MaximumGruen>

<MinimumBlau>**0**</MinimumBlau>

<MaximumBlau>**1**</MaximumBlau>

</Farbe>

«MinimumRot» legt fest, welcher Rot-Wert das Licht mindestens haben muss.

Erlaubte Werte:

0.0000 – 1.0000 - je höher der Wert desto höher der minimale Rot-Anteil

«MaximumRot» legt fest, welcher Rot-Wert das Licht höchstens haben darf.

Erlaubte Werte:

0.0000 – 1.0000 - je höher der Wert desto höher der maximale Rot-Anteil

«MinimumGruen» legt fest, welcher Grün-Wert das Licht mindestens haben muss.

Erlaubte Werte:

0.0000 – 1.0000 - je höher der Wert desto höher der minimale Grün-Anteil

«MaximumGruen» legt fest, welcher Grün-Wert das Licht höchstens haben darf.

Erlaubte Werte:

0.0000 – 1.0000 - je höher der Wert desto höher der maximale Grün-Anteil

«MinimumBlau» legt fest, welcher Blau-Wert das Licht mindestens haben muss.

Erlaubte Werte:

0.0000 – 1.0000 - je höher der Wert desto höher der minimale Blau-Anteil

«MaximumBlau» legt fest, welcher Blau-Wert das Licht höchstens haben darf.

Erlaubte Werte:

0.0000 – 1.0000 - je höher der Wert desto höher der maximale Blau-Anteil

# Stegtransparenz

* PNG-Alpha-Channel
* Alpha-Channel

Wie in Programm=? Was ist alpha-Channel? Weshalb PNG?

# Rotation der Lichter

Orientierung wo? Mathematisch an welcher Achse? Z-Achse

# Drehende Textur

## Umgang mit Renderer

Wie per Programm?

Wie ist Parameter anzusehen?

# UV Mapping

Was ist eine UV-Map?

Wie wird sie verwendet?

Wie sieht das im Code aus?

# SketchUp Modell zu Unity

Datentyp

Einheit-> Inch

Kreis-Segmente (24 -> 90) = Runder

# Transformation Elemente

Vertex-Transformationen

Siehe Einheit import Unity -> 1m

Transformation in Z-Achse = Länge

Transformation XY = Durchmesser / Breite

Code-Beispiel?

# XML-Mapping

XML-Format auf Klassen ummünzen

# Best Practices VR

Occoulos Best Practices -> Aufgabenstellung

# Deployment

# Ausblick

* Implementation Kurve
* Kurve -> Beliebig (nicht nur 90°)
* Muster der Lichtpunkte
* Physikalischer Steg (wissenschaftlicher bericht)