Systemspezifikation Vortex-Tunnel

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

Inhaltsverzeichnis

[Einführung 4](#_Toc502692832)

[Anforderungen 4](#_Toc502692833)

[Abgrenzung der Anforderungen 5](#_Toc502692834)

[Use Cases 6](#_Toc502692835)

[Use Case Diagramm 6](#_Toc502692836)

[Use Case: Konfigurieren 6](#_Toc502692837)

[Use Case: Tunnel durchlaufen 6](#_Toc502692838)

[Allgemeine Entwurfsentscheide 8](#_Toc502692839)

[Effektgenerierung der Drehung 8](#_Toc502692840)

[Prototyp Pointlight 8](#_Toc502692841)

[Prototyp Spotlight 8](#_Toc502692842)

[Prototyp Textur 8](#_Toc502692843)

[Vergleich der Prototypen 9](#_Toc502692844)

[Varianten zur Parameter-Eingabe 10](#_Toc502692845)

[Variante 1: Eingabemaske beim Start der Tunnel-Simulation 10](#_Toc502692846)

[Variante 2: Parameter in einem XML-File 10](#_Toc502692847)

[Variante 3: Ein simpler Editor zum Tunnelbau 10](#_Toc502692848)

[Vergleich der drei Varianten 10](#_Toc502692849)

[Konfigurierbarkeit Steg 12](#_Toc502692850)

[Wahl der Tunneltextur 12](#_Toc502692851)

[Tunnelgenerierung 13](#_Toc502692852)

[Systemübersicht 15](#_Toc502692853)

[Komponentendiagramm 16](#_Toc502692854)

[Hardware-Anforderungen 16](#_Toc502692855)

[Schnittstellen 17](#_Toc502692856)

[Konfigurations-Datei 18](#_Toc502692857)

[Struktur 18](#_Toc502692858)

[Vortex-Tunnel 19](#_Toc502692859)

[Abschnitte 19](#_Toc502692860)

[Abschnitt 20](#_Toc502692861)

[Wandmuster 20](#_Toc502692862)

[Textur 21](#_Toc502692863)

[Lichter 22](#_Toc502692864)

[Farbe 23](#_Toc502692865)

[Stegtransparenz 24](#_Toc502692866)

[Rotation der Lichter 25](#_Toc502692867)

[Drehende Textur 26](#_Toc502692868)

[Umgang mit Renderer 26](#_Toc502692869)

[UV Mapping 27](#_Toc502692870)

[SketchUp Modell zu Unity 28](#_Toc502692871)

[Transformation Elemente 29](#_Toc502692872)

[XML-Mapping 31](#_Toc502692873)

[Best Practices VR 32](#_Toc502692874)

[Deployment 33](#_Toc502692875)

[Ausblick 34](#_Toc502692876)

# 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 aufgrund einer XML-Konfiguration generiert werden
* Versuchsbeschreibung zur Effekterreichung mit Auswertung über Fragebögen   
  (mindestens 10 Datensätze)

Kann:

* Der Tunnel soll Kurven aufweisen

Aufgrund der Zwischenpräsentation der Prototypen wurde folgende Muss-Anforderung zusätzlich formuliert (siehe Anhang «20171108\_Protokoll Zwischenpräsentation Vortex-Tunnel»)

Muss:

* In der XML-Konfiguration sollen zusätzlich die Segmente des Tunnels definiert werden können

Basierend auf den Erkenntnissen aus der Versuchsdurchführung wurde an der Präsentation der Versuchsresultate 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 (siehe Anhang «20171207\_Protokoll Versuchsresultate Vortex-Tunnel»):

* + ***Handgeländer ein/ausschaltbar***
  + ***Breite des Stegs***
  + ***Transparenz-Grad des Stegs***
  + ***Textur***
  + ***Höhe***

## Abgrenzung der Anforderungen

Mittels eines Workshops mit Herr Biallas (siehe Anhang «20171011\_Protokoll Sitzung Vortex-Tunnel») wurden folgende Einschränkungen an der Aufgabenstellung beschlossen:

* Kurven sollen nur im 90° Grad Winkel konzipiert werden
* Lichtmuster sind nicht Teil der Konfiguration und können zufällig sein
* Form der Lichtpunkte ist nicht Teil der Konfiguration und kann beliebig sein
* Tunnel ist immer ein Kreis / Zylinder, eine Ellipse wird explizit ausgeschlossen
* Toleranz der Grössenunterschiede zwischen Konfigurationsdatei und Vortex-Tunnel beträgt 10cm

## 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 Kapitel «UV Mapping»).

Um die Textur zu drehen wurden folgende Wege untersucht:

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

Eine Drehung des Modells ist solange möglich, bis es an Kurven angewendet werden soll. Wie in Abbildung 1 Problemfall Modelldrehung sichtbar wird dreht sich das Kurvenstück (rot) vom anschliessenden Tunnel-Teilstück weg und der Tunnel bekommt ein Loch.

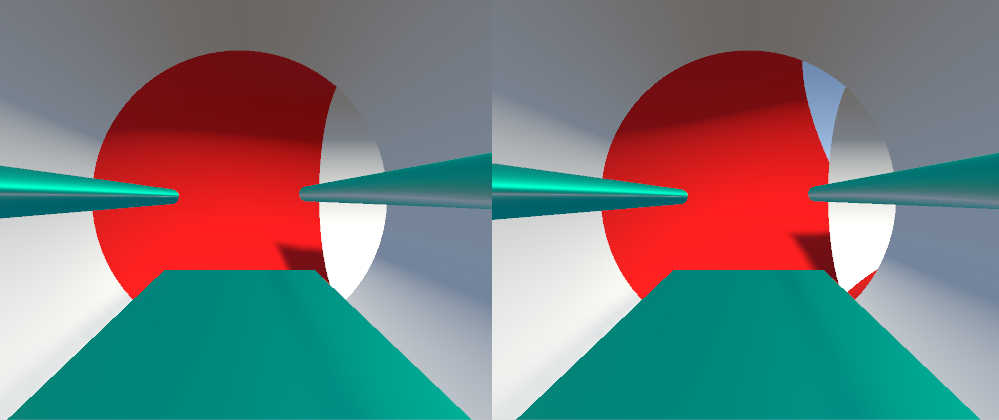


Abbildung 1 Problemfall Modelldrehung

Die Drehung der Kamera hat das Problem, dass diese nicht in Zusammenhang mit Kurven verwendet werden kann. Wie in Abbildung 2 Problemfall Kameradrehung zu sehen ist dreht sich der Ausgang (rot) der Kurve mit der Kamera mit und verunmöglicht somit die korrekte Darstellung einer Kurve.

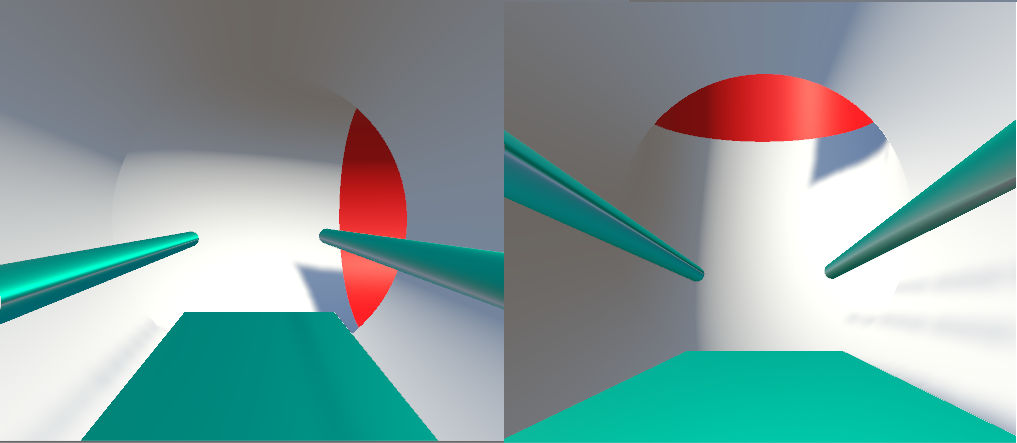


Abbildung 2 Problemfall Kameradrehung

Wird die Drehung über die Textur gelöst, so kann eine Drehung auch in der Kurve erzeugt werden. Dabei wird es jedoch zu Verzerrungen oder Clipping kommen, das genaue Verhalten kann per UV-Map (siehe Kapitel «UV Mapping»). Eine mögliche Lösung ist in «Abbildung 3 Lösungsansatz Texturdrehung» zu sehen.

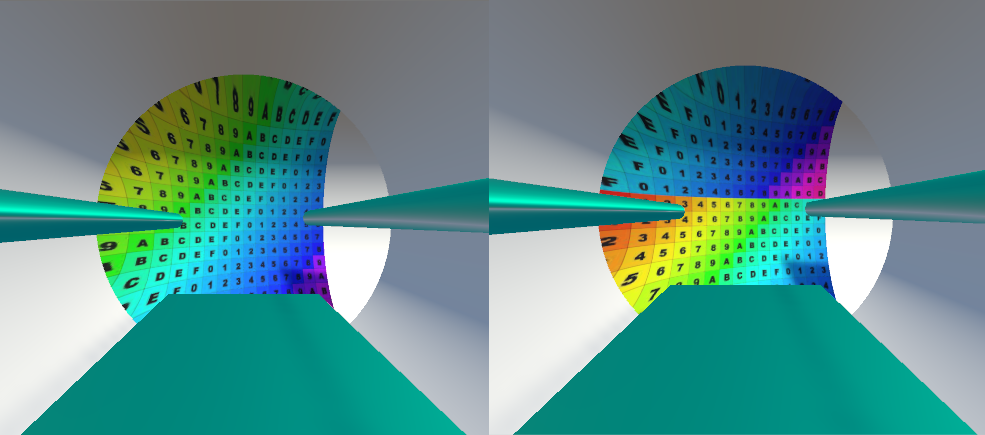


Abbildung 3 Lösungsansatz Texturdrehung

Da die Lösung mittels Drehung der Textur einen Lösungsansatz für das Problem der Drehung der Kurve bietet wurde dieser Lösungsansatz weiterverfolgt.

### Vergleich der Prototypen

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 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

Bei der Erstellung des Prototypen

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

Im Gegensatz zu einem klassischen Software-Entwurf ist bei Unity das Konzept auf sogenannte «Prefabs» und Scripts ausgelegt. Dabei sind die Prefabs immer Objekte welche, sichtbar oder unsichtbar, auf der Szene platziert werden.

Sobald ein Prefab geladen wird werden sämtliche angeschlossenen Scripts ausgeführt. Dazu implementieren sämtliche Scripts das Interface «MonoBehavior» des Imports «UnityEngine».

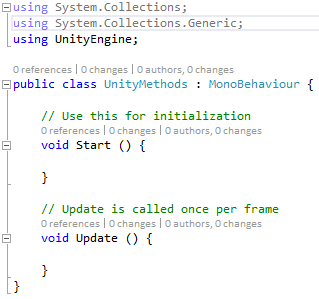


Abbildung 4 Unity-Methoden

Wie in «Abbildung 4 Unity-Methoden» sichtbar ist werden dabei zwei Methoden zur Verfügung gestellt, die Start-Methode welche bei erstmaliger Instanziierung stattfindet und die Update-Methode welche pro Frame ausgeführt wird.

Die Generierung neuer Objekte basierend auf «Prefabs» setzt einen nicht performanten Suchprozess (siehe «Abbildung 5 Instanzierung neues Objekt») und sollte daher möglichst wenig passieren.

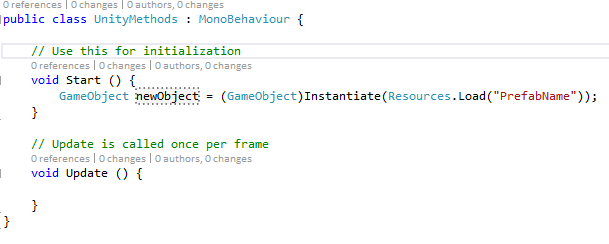


Abbildung 5 Instanzierung neues Objekt

## Komponentendiagramm

Aufgrund des nicht performanten Prozesses zur neuen Objektinitialisierung haben wir uns entschieden, dies mit dem Script «BuildTunnel» zu lösen. Am Anfang der Szene ist nur das Prefab «Tunnel\_Builder» sichtbar welches dann sofort das «BuildTunnel»-Script startet. Dort wird aufgrund der XML-Konfiguration eine oder mehrere Instanzen der Tunnel-Teile instanziiert wie in «Abbildung 6 Komponentendiagramm Vortex-Tunnel» aufgezeigt wird.

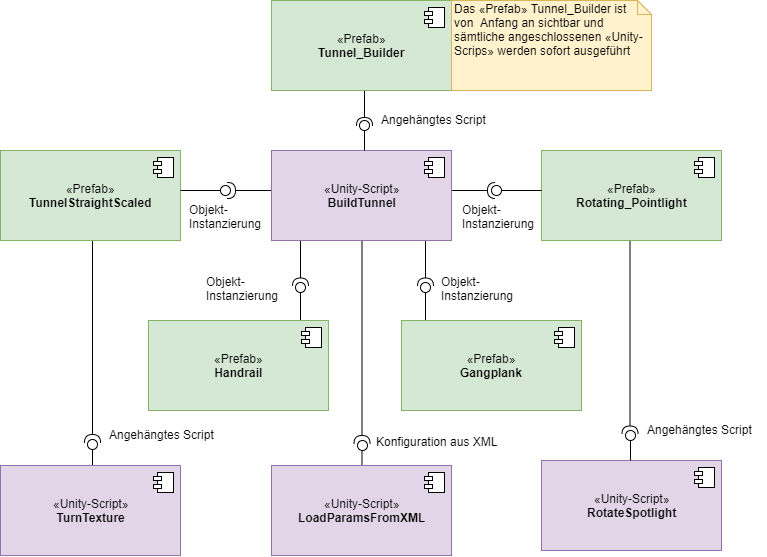


Abbildung 6 Komponentendiagramm Vortex-Tunnel

Aufgrund der Verwendung eines einzelnen Scripts kann so sichergestellt werden, dass alle generierten Objekte zum gleichen Zeitpunkt sichtbar werden. Zusätzlich wird der Ressourcen-Intensive Suchprozess (siehe «Abbildung 5 Instanzierung neues Objekt») nicht durch andere Scripts verlangsamt. Aus Performance-Sicht kann somit die schnellstmögliche Generierung der benötigten Objekte sichergestellt werden. Dies ist insbesondere Wichtig bei einer hohen Anzahl Lichter im Tunnel.

## Hardware-Anforderungen

VR-Computer mit folgenden Eigenschaften:

* nVidia GeForce 970+ / AMD Radeon RX 480 oder besser
* Intel Core i5 / AMD FX 8350 oder besser
* Steam-Account
* Vive-Account
* Stabile Internet-Verbindung

# 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>

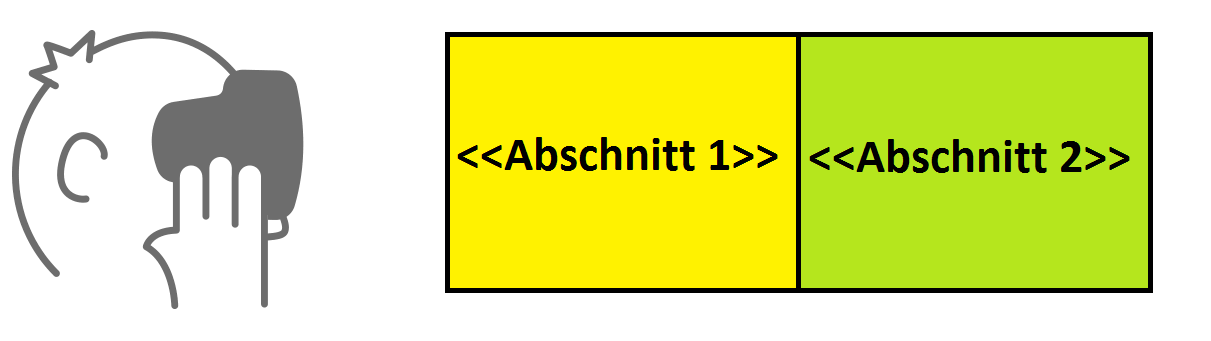
In «**Abschnitte**» werden die einzelnen Tunnel-Abschnitte (siehe weiter unten) spezifiziert. Dabei werden die Tunnel-Abschnitte von oben nach unten abgearbeitet.   


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

Als UV Mapping bezeichnet man den Prozess wie man eine 2D-Textur auf ein 3D-Modell umrechnet. Dabei wird durch das UV-Mapping festgelegt welche Koordinaten der Textur an welche XYZ-Koordinaten des 3D-Modells gelegt werden sollten.

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)