Software Dokumentation
Whack-a-Village

Hochschule Düsseldorf
BMI, BMT, MMI

Florian Kaulfersch
Stefan Böhling
Ben Fischer
Hendrik Schulte
Nanette Ratz
Paul Kretschel

- - -

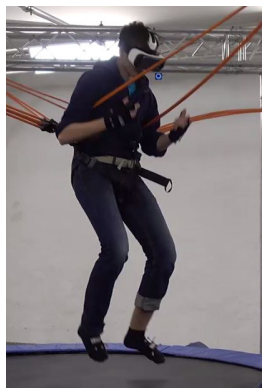
Prof. Dr. Christian Geiger

Hardware Voraussetzungen

Das System besteht aus mehreren Hardware-Komponenten.

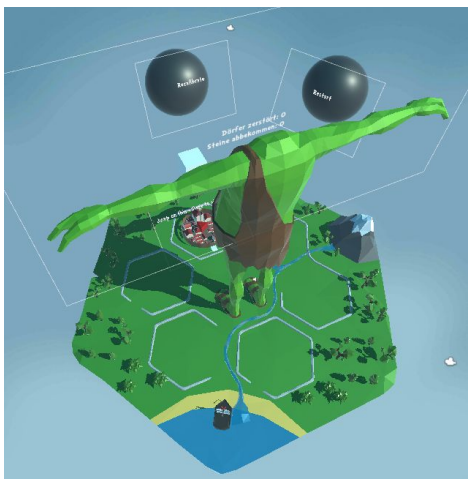
- Samsung Galaxy S6
- Samsung GearVR
- OptiTrack System mit möglichst vielen Kameras
    - Marker für Hände, Füße und Kopf
- Ein Windows Desktop als OptiTrack-Server
- Ein ca. 4x4x3 Meter Traversenkäfig
- Ein Garten-Trampolin
- Sicherungsgurte





Architektur

Das Projekt wurde in Unity3D 5.5.0f3 entwickelt und für Android 6.0.1 gebuildet. In der Scene *HenneScene.unity* befinden sich alle wichtigen Objekte. Die verschiedenen Bereiche der Software sind auf einzelne Root-GameObjects aufgeteilt.




GameManager

Der GameManager steuert den Spielablauf und kontrolliert die Erstellung von Spielobjekten, wie Dörfern Katapulten und Zeppelinen.

UserContainer

Dieses Objekt beinhaltet den trolligen Spielercharakter, sowie dessen Steuerung und Anbindung an das OptiTrack-System. Dieser Code stammt ursprünglich aus dem Projekt Superjump, auf welchem dieses Projekt aufbaut und wurde an unsere Bedürfnisse angepasst. Unter dem Unterpunkt Tracking befinden sich die unsichtbaren Rigidbodies, mit denen der Spieler mit der Welt interagieren kann.

UI

Hier befinden sich die Objekte für verschiedene HUD-Anzeigen. Dazu zählen Kalibrierungshinweise zu Beginn des Spiels, sowie ein Indikator für den Spielfortschritt.

MainMenu / TestUI

Dieses GameObject beinhaltet das 3D Benutzerinterface, aus dem das Hauptmenü zusammengesetzt wurde. Es beinhaltet mehrere schwarze Kugeln und ein Dorf auf dem Boden, die als 3D-Buttons dienen. Unter TestUI ist ein *Interactor* genannter Würfel, mit dem im Editor die Interaktion des Spielers simuliert werden kann.

Spielelemente

Alle restlichen Spielobjekte sind auf die folgenden Bereiche aufgeteilt:

BottomModels

Hierzu zählt die Insel, auf der der Spielercharakter steht, zusammen mit allen Objekten, die sie bevölkert. Die Insel gibt nach unten nach, wenn der Spieler den Boden berührt, und springt zurück an seine Ausgangsposition, wenn er in die Luft springt. Damit soll das Nachgeben des Trampolins in die Virtuelle Welt übertragen werden. Dieser Effekt wird durch das anhängende Script *Jump Feedback* kontrolliert.

EnvironmentalModels

Diese Objekte zählen zum Hintergrund und sorgen für Atmosphäre und vermitteln dem Spieler ein Gefühl für seine Größe. Es sind fliegende Inseln und Wolken.

GameModels

Dieses Objekt ist für alle fliegenden Objekte gedacht, die nicht an die Bwegung der Insel gekoppelt sind. Zur Zeit beinhaltet es nur den Zeppelin.

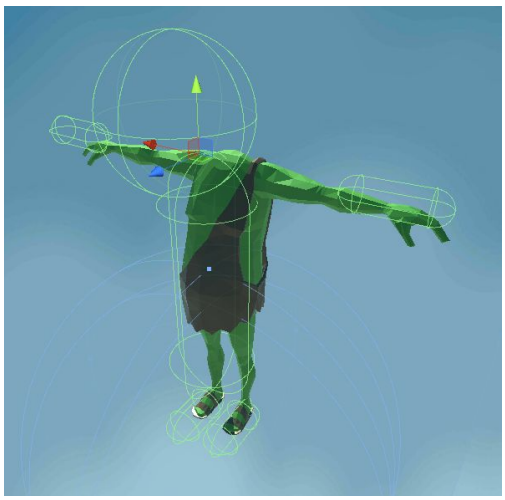
AmbientSoundManager

Dieses Objekt sorgt für die akustische Hintergrundbeschallung. Dazu zählt zum einen ein dauerhafter ambienter Sound, der in einer Schleife abgespielt wird. Außerdem Gibt es einige Einspieler für den Troll, die zwischendurch in unregelmäßigen Abständen abgespielt werden.

Tracking und Inverse Kinematics

Es werden zwei verschiedene Tracking Verfahren kombiniert, um die Steuerung des Spiels zu ermöglichen. Die Rotation des Kopfes wird durch das auf Intertialsensoren besierendem System der GearVR mit dem Galaxy S6 erfasst. Zusätzlich trackt das OptiTrack-System die absolute Position der Hände, Füße und des Kopfes des Spielers. Die Daten werden an den OptiTrack-Server geschickt, an dem sie verarbeitet und über WLan verteilt werden. Dafür wird die zugehörige Software Motive verwendet. Bei Start des Spiels ruft das Smartphone dann die Trackingdaten ab und errechnet über Inverse Kinematics die Bewegungen des Spielercharakters. Dieser Teils des Programms wurde von Daniel Kirchhof, Okkan Köse, Marces Tiator und Roman Wiche unter dem Namen Superjump entwickelt.

Interaktion



Die Interaktion des Spielers wird hauptsächlich über physische Bewegungen erreicht. Jeder Tracker in der realen Welt hat ein entsprechendes Gegenstück in der virtuellen Welt. Dieses Bewegt sich unabhängig vom Modell des Trolls, welches nur versucht, sich möglichst gut an die tatsächliche Position der Extremitäten anzupassen. Die virtuellen Tracking Objekte befinden sich unter UserContainer > Tracking. Jedes Objekt liegt auf dem Layer *Player*, besitzt einen Collider, eine Rigidbody und ein Script namens *Force Applicator*. Dieses Script sorft dafür, dass das Objekt Events auslösen kann, wenn Spielelemente berührt werden. Außerdem generiert es aus den Positionen in vorherigen Frames die Geschwindigkeit, mit der der Spieler sich bewegt hat, und gibt sie an das Spielobjekt weiter. Als Gegenstück dazu dienen die Scripts *RealWorldButton* und *Destructible*. RealWorldButton steuert die 3D Buttons im Hauptmenü, während Destructible als Basis für verschiedene Spielobjekte gillt. Diese Klasse wird von den Dörfern, Katapulten und Zeppelinen verwendet und reagiert, wenn der Spieler mit dem Objekt in Berührung kommt. Wenn die Geschwindigkeit, die der Spieler bei der Berührung hat, den *Force Threshhld* überschreitet, so wird das Objekt zerstört, ein Partikeleffekt erzeugt und ein Sound abgespielt. Dem Script kann mit der Funktion *AddOnDestroyListener* eine Aktion mitgeben werden, die bei der Zerstörung ausgelößt werden soll.

Darüber hinaus können mit der Klasse *SwipeDetectorGear* auf die Touchsensitiven Buttons der GearVR benutzt werden. Dies geschiet über die Methoden AddSwipeTopListener.

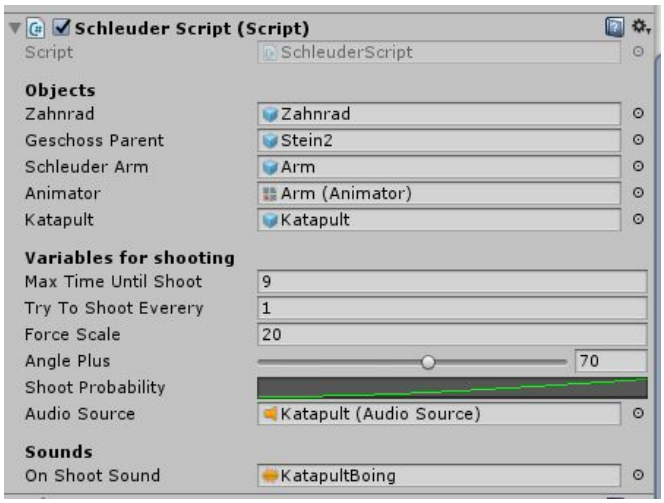
Gameplay

Spielablauf

Der Spielablauf wird im GameManager. gesteuert. Hier werden alle Nachrichten des Hauptmenüs verarbeitet und entsprechende Aktionen ausgeführt. Sobald das Spiel gestartet wurde, wird eine Couroutine gestartet, die in kurzen Zeitabständen zufällig ein Dorf oder ein Katapult an einem der Bauplätze erstellt. Die Bauplätze (*BuildingSites*) werden als GameObject unter BottomModels > BuildingSites definiert.

Katapult System

Für die Katapult - Gamelogic muss die Komponente “SchleuderScript” auf dem Katapult-Gameobject liegen.



Objects

Zahnrad	Zahnrad GameObject (Kindobjekt des Katapults)
Geschoss Parent	Stein-Gameobject , das die Komponenten RigidBody, Sphere Collider und SteinScript enthält (Stein Prefab das abgeschossen wird)
Schleuder Arm	Arm-Objekt des Katapults
Animator	Animator Komponente des Arms
Katapult	Katapult Gameobject

Variables for shooting

Max time until shoot	Zeit in Sekunden die das Katapult maximal ruht ohne zu feuern. Sollte den gleichen Wert wie die X-Achse der ShootProbability-Curve besitzen.
Try to shoot every	Intervall in Sekunden in welchem das Katapult, abhängig von der Kurve ShootProbability, versucht zu feuern
Force Scale	Verstärkung der Schusskraft. (Empfohlen: 20)
Angle Plus	Winkel in Grad mit dem der Vektor “Katapult->Spieler-Kopf” um die Z-Achse rotiert wird (= Höhe des Schusses)
ShootProbability	AnimationCurve. X-Achse: Zeit (Sollte die Länge Max Time until shoot besitzen), Y-Achse: Wahrscheinlichkeit des Feuerns (0.0 - 1.0)
Audio Source	Quelle des Schuss-Sounds

Visualisierung

AgentSpawner

Dieses System erzeugt kleine Agenten, die sich während des Spiels zwischen den Dörfern bewegen, um einen lebendigeren Eindruck zu verschaffen. Außerdem soll es dem Spieler seine eigene Größe vermitteln.

Spin

Dieses Script steuert die Bewegung der Hintergrundobjekte und lässt sie langsam um den Spieler oder sich selbst rotieren. Damit soll die Umgebung lebendiger wirken.