HTBLVA, Wien V, Spengergasse

Höhere Lehranstalt für Medien
Ausbildungsschwerpunkt Gamedesign

DIPLOMARBEIT

Gesamtprojekt

Tricks 'n' Treats - Entwicklung eines asymmetrischen VR- und Couch-Partyspiels

Technische Implementierung und Optimierung von asymmetrischen VR-Spielen.
Simon Buketits 5AHMNG Betreuer: Kristian Ljubek, BSc MSc

Design eines asymmetrischen Local-Multiplayer-Party-Games unter Verwendung des MDA-Frameworks
Felix Kaspar 5AHMNG Betreuer: René Ksuz, BSc MA

Farb- und Formpsychologie in asymmetrischen VR-Spielen
Emma Wehinger 5AHMNG Betreuer: Alexander Hager

Performance-Optimierung von 3D Assets in VR-Spielen
Ian Memic 5AHMNG Betreuer: Alexander Hager

Schuljahr 2022/23

Abgabevermerk:

Datum: 31.03.2023 *übernommen von:*

Dokumentenversion: HTL-RDP_2017-1

SPENGERGASSE	*HÖHERE TECHNISCHE BUNDESLEHRANSTALT SPENGERGASSE*
	Fachrichtung: **Design** *Ausbildungsschwerpunkt:* *Gamedesign*

DIPLOMARBEIT
DOKUMENTATION

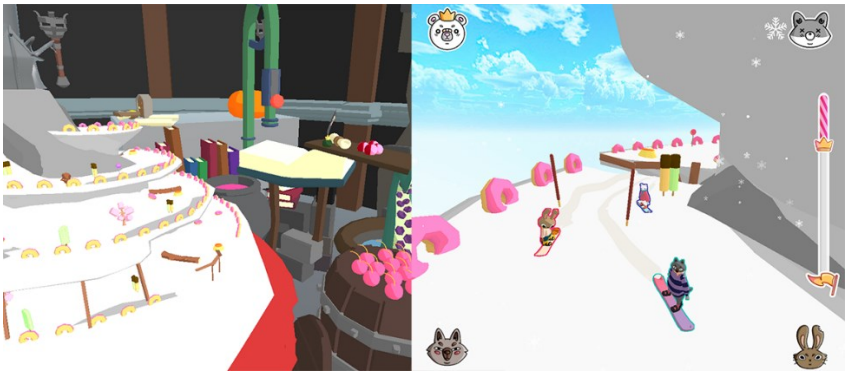
Namen der Verfasser/innen	*Simon Buketits* *Felix Kaspar* *Emma Wehinger* *Ian Memic*
Jahrgang *Schuljahr*	*5AHMNG* *2022/23*
Thema der Diplomarbeit	*Tricks 'n' Treats - Entwicklung eines asymmetrischen VR- und Couch-Partyspiels*
Kooperationspartner	*CYBERTIME GmbH* *(E-Mail: office@cybertime.at, https://cybertime.at/)* *Mariahilfer Straße 113, A-1060 Wien, Raffael Leb,* *+43 1 718 55 00 50*

Aufgabenstellung	*Das Untersuchungsanliegen der Arbeit ist es, ein Spiel zu entwickeln, dass die Welten von Couchparty und VR glaubhaft verschmilzt und eine chancengleiche Experience für alle bietet. Um das zu erzielen, sollen alle Teilarbeiten einen wichtigen Beitrag leisten. Es soll Teamwork und Rivalität erzeugt werden, eine ausgeglichene Atmosphäre und eine Stabilität und Performance, die das Spiel unter den gegebenen Voraussetzungen spielbar macht.*

Realisierung	*Ziel ist es, eine interessante Dynamik zwischen einem Team von Couch Party-SpielerInnen und eines Spielers in VR zu schaffen, um das noch neue Konzept von einem Mix zwischen VR und Couchparty genauer zu untersuchen. Außerdem sollen somit VR-Spiele einer neuen Zielgruppe zugänglicher gemacht werden und dabei dem Nachteil von VR, der geringen Verbreitung des Mediums, entgegenwirken.*

Ergebnisse	*Man kann nicht alles vorhersagen, und selbst ein gut designtes Spiel braucht weitere Anpassungen, um die gewünschten Ergebnisse zu erzielen. Nach einigen Anpassungen kam das Spiel sehr gut bei den Testenden an. Sowohl die allgemeine Atmosphäre als auch die Experience und Performance des Spieles waren auf einem sehr guten Niveau. Vor allem die Interaktionen zwischen den PC-Spielenden und dem VR-Spielenden zeigten, dass alle in der Spielwelt investiert waren und mitgefiebert haben, um zu gewinnen.*

SPENGERGASSE	*HÖHERE TECHNISCHE BUNDESLEHRANSTALT SPENGERGASSE*
	Fachrichtung: ***Design*** *Ausbildungsschwerpunkt:* *Gamedesign*

Typische Grafik, Foto etc. (mit Erläuterung)	 *Eine typische Spielsituation im Projekt wird durch die obere Grafik dargestellt. Der linke Teil der Grafik zeigt, die Welt aus der Sicht des VR-Spielenden und wie versucht wird einen Zauber auf die Snowboarder zu werfen, um sie besiegen. Der Rechte Teil der Grafik zeigt das Spiel aus der Sicht der Snowboarder, und wie sie versuchen, den durch den Magier platzierten Hindernissen auszuweichen und entgegenzusteuern. In den Ecken wird ihr derzeitiger Status (Krone, Lebendig, Tot) dargestellt.*

Teilnahme an Wettbewerben, Auszeichnungen	*Nein*

Möglichkeiten der Einsichtnahme in die Arbeit	*Ja*

Approbation *(Datum / Unterschrift)*	*Prüfer/Prüferin*	*Direktor/Direktorin* *Abteilungsvorstand/Abteilungsvorständin*

SPENGERGASSE	HÖHERE TECHNISCHE BUNDESLEHRANSTALT, WIEN V, SPENGERGASSE
	Department: **Design** Educational focus: **Gamedesign**

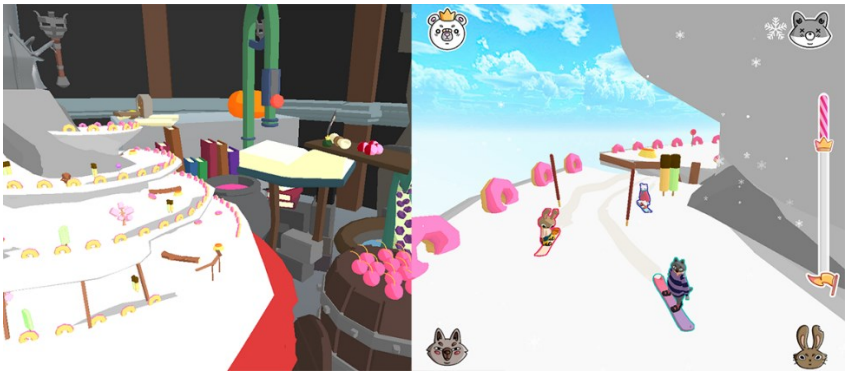
DIPLOMA THESIS
Documentation

Author(s)	Simon Buketits Felix Kaspar Emma Wehinger Ian Memic
Form Academic year	5AHMNG 2022/2023
Topic	Tricks 'n' Treats - Entwicklung eines asymmetrischen VR- und Couch-Partyspiels
Co-operation partners	CYBERTIME GmbH (E-Mail: office@cybertime.at, https://cybertime.at/) Mariahilfer Straße 113, A-1060 Wien, Raffael Leb, +43 1 718 55 00 50

Assignment of tasks	The research goal of the thesis is to develop a game that merges the worlds of couch party and VR in a credible way and offers an equal opportunity experience for all. In order to achieve this, all sub-works should make an important contribution. Teamwork and rivalry should be created, a balanced atmosphere and a stability and performance that makes the game playable under the given conditions.

Realisation	The goal is to create an interesting dynamic between a team of couch party players and a player in VR in order to explore the still new concept of a mix between VR and couch party in more detail. It also aims to make VR games more accessible to a new target audience, counteracting the disadvantage of VR, the low market penetration of the medium.

Results	One can't predict everything, and even a well-designed game needs further adjustments to achieve the desired results. After some adjustments, the game was very well received by the testers. The general atmosphere as well as the experience and performance of the game were on a very good level. In particular, the interactions between the PC players and the VR player showed that everyone was invested in the game world and rooting to win.

Illustrative graph, photo (incl. explanation)	 *A typical game situation in the project is represented by the upper graphic. The left part of the graphic shows the world from the VR player's point of view and how they try to cast a spell on the snowboarders to defeat them. The right part of the graphic shows the game from the snowboarders' point of view, and how they try to dodge and counter the obstacles placed by the mage. In the corners their current status (Crown, Alive, Dead) is shown.*

Participation in competitions Awards	*No*

Accessibility of diploma thesis	*Yes*

	Examiner	*Head of College / Department*
Approval (date / signature)		

“Tricks ’n’ Treats“: Entwicklung eines asymmetrischen VR- und Couch-Partyspiels

Ian Memic, Simon Buketits, Emma Wehinger und Felix Kaspar

DIPLOMARBEIT

eingereicht am
Ausbildungsschwerpunkt
Mediendesign - Gamedesign
in Wien

im März 2023

© Copyright 2023 Ian Memic, Simon Buketits, Emma Wehinger und Felix Kaspar

Diese Arbeit wird unter den Bedingungen der *Creative Commons Lizenz Namensnennung–NichtKommerziell–KeineBearbeitung Österreich* (CC BY-NC-ND) veröffentlicht – siehe http://creativecommons.org/licenses/by-nc-nd/3.0/at/.

Erklärung

Ich erkläre eidesstattlich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen nicht benutzt und die den benutzten Quellen entnommenen Stellen als solche gekennzeichnet habe. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt.

Wien, am 31. März 2023

Ian Memic, Simon Buketits, Emma Wehinger und Felix Kaspar

Inhaltsverzeichnis

Erklärung **iii**

Kurzfassung **x**

Abstract **xi**

Technische Implementierung und Optimierung von asymmetrischen VR-Spielen **1**

1 Einleitung **2**
1.1 Virtuelle Realität . . . 2
1.1.1 Asymmetrische VR-Spiele . . . 2
1.2 Warum asymmetrisches VR? . . . 3
1.3 Zielsetzung . . . 3

2 Überblick über die Virtuelle Realität **4**
2.1 Verfügbare VR-Brillen . . . 4
2.2 Existierende Spiele . . . 5
2.2.1 “Keep Talking and Nobody Explodes“ . . . 5
2.2.2 “Davigo“ . . . 5
2.2.3 “Carly and the Reaperman“ . . . 5
2.3 Erwartete Probleme . . . 6
2.3.1 Motion-Sickness und VR-Krankheit . . . 6
2.3.2 Was ist die Frame-Rate? . . . 7
2.3.3 Wenige Bilder pro Sekunde . . . 7
2.3.4 Platz und Bewegung . . . 7
2.3.5 Interaktionen . . . 7

3 Mögliche Performance-Optimierungen **8**
3.1 Level Of Detail . . . 8
3.2 Culling . . . 9
3.2.1 Beispiel . . . 9
3.2.2 Frustum-Culling . . . 9
3.2.3 Backface-Culling . . . 10

3.3 Draw-Calls . . . 10
3.3.1 Dynamic-Draw-Call-Batching . . . 10
3.3.2 Static-Draw-Call-Batching . . . 11
3.4 Object-Pooling . . . 11
3.5 Coroutines vs Update . . . 12
3.6 Statische -und Dynamische Collider . . . 12
3.6.1 Static-Collider . . . 12
3.6.2 Rigidbody-Collider . . . 13
3.6.3 Kinematic-Rigidbody-Collider . . . 13

4 Umsetzung der PC- und VR-Welt 14
4.1 Levelaufbau . . . 14
4.2 VR-Charakter . . . 15
4.2.1 Zauber -und Interaktionssystem . . . 15
4.2.2 VR-Umgebung . . . 15
4.3 PC-Charakter . . . 15
4.3.1 Charakter-Größe . . . 15
4.3.2 Fähigkeiten . . . 16
4.4 Lösung der erwarteten Probleme . . . 16
4.4.1 Wenige Bilder pro Sekunde . . . 16
4.4.2 Level Of Detail . . . 16
4.4.3 Culling . . . 16
4.4.4 Collider . . . 17
4.4.5 Object-Pooling . . . 17
4.4.6 Drawcall-Batching . . . 17
4.4.7 Bewegungskrankheit und Rotationsproblem . . . 17
4.4.8 Platz und Bewegung . . . 18
4.4.9 Interaktionen . . . 19

5 Fazit 20

Design eines asymmetrischen Local-Multiplayer-Party-Games unter Verwendung des MDA-Frameworks 21

6 Begriffsdefinitionen 22
6.1 VR trifft Couchparty-coop . . . 22
6.1.1 Was ist Virtual Reality? . . . 22
6.1.2 Was ist Couchparty? . . . 23
6.1.3 Was sind coop-games? . . . 24
6.1.4 Was bedeutet Asymmetrie? . . . 25

7 Spaß in Videospielen 26
7.1 Warum machen Spiele Spaß? . . . 26
7.1.1 Woher kommt Spaß? . . . 26

7.1.2 Spaß bei mehrspieler Spielen . . . 26
7.2 Wie kann man Spaß „kreieren“? . . . 27
7.2.1 MDA-Framework . . . 28
7.2.2 Essential Experience . . . 29
7.2.3 Core Mechanics . . . 29
7.2.4 Pacing, Flow . . . 29
7.2.5 Player Action Feedback . . . 30

8 Stand der Industrie **31**
8.1 Asymmetrical Coop . . . 31
8.1.1 Keep Talking and Nobody Explodes . . . 31
8.1.2 It Takes Two . . . 31
8.2 Couchparty . . . 31
8.2.1 Mario Party . . . 31
8.2.2 Jackbox . . . 32
8.3 Couchparty VR-Games . . . 32
8.3.1 Acron: Attack of the Squirrels! . . . 32
8.3.2 Takelings House Party . . . 32

9 Implementation **33**
9.1 Was muss bei VR-Game-Design beachtet werden? . . . 33
9.2 Stimmung des Spieles . . . 33
9.2.1 Core Pillars . . . 33
9.2.2 Target Experiences . . . 34
9.3 Zusammentreffen beider Welten (PC + VR) . . . 35
9.3.1 Streckendesign . . . 35
9.3.2 Interaktionsdesign . . . 36
9.3.3 Playercontroller & Obstacles . . . 38
9.3.4 Teamwork erzeugen - Krone & andere Lösungsansätze 40

10 Post-Mortem **41**
10.1 Playtests . . . 41
10.2 Kommunikation . . . 42
10.3 Zukunft des Projektes . . . 43

Farb- und Formpsychologie in asymmetrischen VR-Spielen **45**

11 Überblick über die Materie **46**
11.1 Farbpsychologie . . . 46
11.1.1 Was ist Farbe? . . . 46
11.1.2 Der Farbkreis und Farbschemas . . . 48
11.1.3 Farbtemperatur . . . 50
11.1.4 Farbrelativität . . . 51
11.1.5 Farbsättigung . . . 51

11.1.6 Farbe, Licht und Schatten . . . 51
11.1.7 Wie wirkt sich Farbe auf unsere Stimmung aus? . . . 52
11.2 Formpsychologie . . . 52
11.2.1 Gestalt und Form . . . 53
11.2.2 Linien und Linien- Stile . . . 53
11.2.3 Kompositionslinien . . . 54
11.2.4 Wie Formen und Linien Emotionen erzeugen . . . 55
11.3 Environment Design . . . 55
11.3.1 Character und Environment Shapes . . . 56
11.3.2 Character Centric Environment Design . . . 56

12 Möglichkeiten zur Implementierung 58
12.1 Nutzung von Farben in Spielen . . . 58
12.1.1 Beispiel – Outlast . . . 58
12.1.2 Beispiel – Journey . . . 59
12.2 Nutzung von Formen in Spielen . . . 60
12.2.1 Beispiel – Journey . . . 62
12.2.2 Beispiel – Super Mario Galaxy . . . 62

13 Konkrete Implementierung 64
13.1 Konzeption der Charaktere . . . 64
13.2 UI/UX Design . . . 67
13.3 Environment Design und Komposition . . . 69
13.4 Andere 2D-Assets . . . 71
13.5 Fazit . . . 71

Performance-Optimierung von 3D Assets in VR-Spielen 73

14 Überblick über die Materie 74
14.1 Performance Optimierung . . . 74
14.1.1 Was ist schlechte Performance? . . . 74
14.1.2 Was löst schlechte Performance aus? . . . 74
14.2 Rendering & die VR Kamera . . . 75
14.2.1 Rendering . . . 75
14.2.2 VR-Kamera . . . 76
14.3 Mesh Optimierung . . . 77
14.3.1 Das Mesh . . . 77
14.3.2 Geometrie vs. Topologie . . . 77
14.3.3 Tris, Quads und N-Gone . . . 80
14.3.4 Die Wichtigkeit von Performance Optimierung in Video Spielen . . . 81
14.3.5 Beispiel: Cyberpunkt 2077 . . . 81
14.3.6 Fragestellung . . . 83
14.3.7 Mesh Resolution . . . 83

14.3.8 Mesh Cleanup 84
14.3.9 Textur Auflösung und UV-Maps 84
14.3.10 Texture Atlasing 87
14.3.11 Billboarding 87
14.3.12 Shading 88
14.3.13 Flat Shading 89
14.3.14 Smooth Shading 89
14.3.15 Detail, ohne auf Performance zu vergessen 90
14.3.16 Texture Baking 92

15 Theoretische Implementation in Tricks ´n´ Treats 94
15.1 Bedenken bei der Implementation 94
15.1.1 Einschränkung durch den Artstyle 94
15.1.2 Unbedingt Notwendig 94
15.1.3 Hat die Methode einen Sinn? 95
15.1.4 Texturing in Tricks ´n´ Treats 96
15.1.5 Texture Baking als Alternative 97

16 Praktische Implementation in Tricks ´n´ Treats 98
16.1 Endresultat 98
16.1.1 Entwicklung des Arbeitsprozesses 98
16.2 Erwartete Ergebnisse 98
16.2.1 Mesh Cleaning in Tricks ´n´ Treats 98
16.2.2 Smooth Shading in Tricks ´n´ Treats 100
16.3 Texturieren – Die Evolution 101
16.3.1 Textur Paletten 102
16.4 Rückblick 103

17 Conclusio 104

A Projektdokumentation 106
A.1 Meilensteine 106
A.1.1 Prototype (23.09) 106
A.1.2 Iteration Interim 1 (20.10) 106
A.1.3 Iteration Interim 2 (22.12) 106
A.1.4 Vertical Slice (30.03) 106
A.2 Playtest Protokolle & Ergebnisse 107
A.2.1 Welcher Spell ist der Lustigste? 107
A.2.2 Welcher Spell ist der Unlustigste? 107
A.2.3 In welcher Rolle hattest du mehr Spaß? 107
A.2.4 Gab es Momente wo du stark mitgefiebert hast? . . . 107
A.2.5 Wie würdest du die Gesamterfahrung bewerten? . . . 107
A.2.6 Gab es Momente in denen dir langweilig war? 107
A.3 Begleitprotokolle 107

A.3.1 Arbeitsprotokoll Simon Buketits 108
A.3.2 Arbeitsprotokoll Felix Kaspar 110
A.3.3 Arbeitsprotokoll Emma Wehinger 114
A.3.4 Arbeitsprotokoll Ian Memic 116

B Kooperationsvereinbarung **118**

C Inhalt des Datenträgers **122**
C.1 Arbeit 122
C.2 Projektdateien 122
C.3 Game Design 122
C.4 Presskit 122

Abbildungsverzeichnis **136**

Tabellenverzeichnis **138**

Kurzfassung

Diese Diplomarbeit befasst sich mit der Entwicklung des asymmetrischen lokalen Multiplayer-Spieles Tricks ‘n’ Treats. Der erste Teil der Arbeit befasst sich mit der Entwicklung von asymmetrischen Virtual Reality Spielen, wobei ein besonderer Fokus auf Performance-Optimierung, den aufgetretenen Problemen und technischen Limitationen von asymmetrischen Virtual Reality Spielen. Im zweiten Teil wird genauer bearbeitet, wie sich die Dynamiken zwischen PC und VR bilden, wie man diese beeinflusst und im Falle des Spieles eine gute Interaktion zwischen den beiden konstruieren kann. Der dritte Teil der Arbeit befasst sich mit der Welt und Atmosphäre des Spiels. Besonders widmet er sich wie Farb- und Formgebung die Gefühle und Aktionen der SpielerInnen beeinflussen. Der vierte Teil beschäftigt sich mit dem 3D-Modeling und geht spezifisch auf die Performance Optimierung der 3D-Assets ein. Dabei liegt ein großer Fokus auf der Recherche und Verwendung verschiedener Optimierungs-Methoden und deren Vor- und Nachteile.

Abstract

This thesis is about the development of Tricks 'n' Treats, an asymmetric local-multiplayer game. The first part of the thesis deals with the technical development of asymmetric virtual reality games, with a special focus on performance optimization, the problems that were encountered and technical limitations of asymmetric virtual reality games. The second part discusses how the dynamics between PC and VR are formed, how to influence them and, in the case of the game, how to construct a good interaction between the two worlds. The third part of the work explores the world and atmosphere of the game. In particular, it is devoted to how color and shape design influences the feelings and actions of the players. The last part deals with 3D modeling and specifically addresses the performance optimization of 3D-game-assets. Thereby a big focus is on the research and use of different optimization methods and their advantages and disadvantages.

Technische Implementierung und Optimierung von asymmetrischen VR-Spielen

Simon Buketits

D I P L O M A R B E I T

eingereicht am
Ausbildungsschwerpunkt
Mediendesign - Gamedesign

betreut durch
Kristian Ljubek, BSc MSc

im März 2023

Kapitel 1

Einleitung

1.1 Virtuelle Realität

Die Virtuelle Realität (Virtual Reality, VR) ist eine computergenerierte virtuelle Umgebung, die in Echtzeit berechnet wird. In dieser ist es möglich, sich umzuschauen und je nach Anwendung kann man sich auch Bewegen und beliebig mit Objekten interagieren. Es gibt zwei verschiedene Möglichkeiten Virtual Reality zu erleben. Einerseits gibt es speziell angefertigte Räume, in denen Großbildleinwände angebracht sind, andererseits gibt es auch Head-Mounted-Displays (VR-Brillen, die man aufsetzt), die hier im Fokus stehen werden. In der Virtuellen Realität gibt es eine breit gefächerte Reichweite an Spielen, in denen man verschiedenste Situationen erleben kann. Anfangs waren Spiele für VR nur simple Testräume, in denen man mit Objekten interagieren konnte. Mittlerweile ist es jedoch möglich, riesige Spiele mit einer eigenen Geschichte zu erleben. Die Interaktionen, die damals ein ganzes Spiel ausmachten, sind seit einiger Zeit nur noch die Basis, worauf neue Spiele aufbauen.

1.1.1 Asymmetrische VR-Spiele

Die meisten VR-Spiele kapseln einen von der Außenwelt ab. Sobald man die VR-Brille aufsetzt, befindet man sich in einer neuen, digitalen Welt. Eine Zeit lang war die zwischenmenschliche Interaktion in der virtuellen Realität gar nicht möglich. Seit Neuerem gibt es auch Spiele, die unterstützen, dass VR-Spieler über das Internet miteinander spielen können (Online-Multiplayer). Somit braucht man jedoch zwei HMDs (VR-Brillen), zwei PCs und man kann in der Regel nicht nebeneinander spielen. Asymmetrische VR-Spiele versuchen, reguläre PC-Spiele mit VR-Spielen zu kombinieren. So können z. B. zwei Freunde mit einem PC und einer VR-Brille miteinander spielen. Das funktioniert z. B., indem die zwei Spieler sich in derselben Spielwelt befinden, jedoch unterschiedliche Charaktere steuern. Durch die verschiedenen Steuerungsmöglichkeiten (VR-Controller, Tastatur, Maus

und Gamepad) ergibt sich oft automatisch eine bestimme Rollenverteilung. So könnte der PC-Spieler z. B. einen Menschen spielen, der durch ein Level manövrieren muss und zeitgleich steuert der VR-Spieler einen Riesen, der den Menschen behindern muss, indem er ihm beispielsweise Steine in den Weg legt.

1.2 Warum asymmetrisches VR?

Die größte Stärke und auch Nachteil der VR-Brille ist, dass sie einen von der Außenwelt abkapselt und dafür sorgt, dass man vollständig in die Spielwelt eintreten kann. Online-Multiplayer sorgt dafür, dass mehrere VR-Spieler miteinander spielen und interagieren können, jedoch gibt es für Interaktion und Kooperation zwischen VR-Spieler und PC-Spieler noch kein großes Angebot.

1.3 Zielsetzung

Das Ziel dieser Arbeit ist, die technische Implementierung und Optimierung von asymmetrischen VR-Spielen genauer zu untersuchen und gefundene technische Probleme genauer zu erläutern.

Kapitel 2

Überblick über die Virtuelle Realität

2.1 Verfügbare VR-Brillen

Es gibt mittlerweile viele verschiedene Anbieter für VR-Brillen, die verschieden gute HMDs in verschiedenen Preisreichweiten anbieten. Das Steam Hardware Survey zeigt, dass momentane Marktführer Facebooks Meta (ehemals Oculus) mit der Oculus Quest und der Oculus Quest 2 im Low-Budget Bereich und Valve mit der Valve Index im High-Budget Bereich sind. (Stand Mitte 2022)

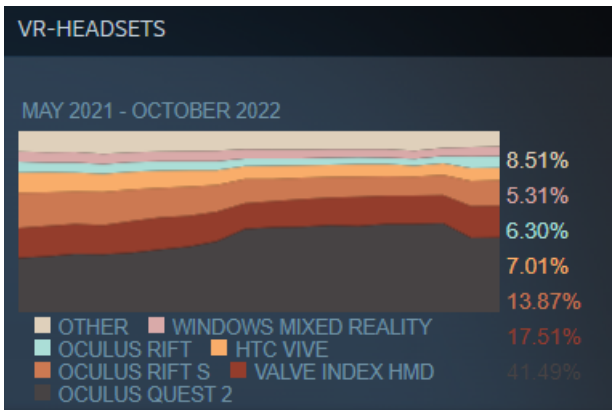


Abbildung 2.1: Ergebnisse des Steam Hardware Surveys aus dem Jahr 2022[1]

2.2 Existierende Spiele

Es gibt viele verschiedene VR-Spiele und eine breite Reichweite an Genres. Besonders relevant sind jedoch die asymmetrischen Spiele. Einige wichtige Beispiele sind hier aufgelistet.

2.2.1 “Keep Talking and Nobody Explodes“

„Du bist alleine in einem Raum mit einer tickenden Zeitbombe gefangen. Deine Freunde haben das Handbuch, um die Bombe zu entschärfen, können sie aber nicht sehen, also müssen sie dir erklären, wie du sie entschärfen kannst.“ [2]

2.2.2 “Davigo“

„DAVIGO“ ist ein VR vs. PC "realitätsübergreifendes"Kampfspiel. Der VR-Spieler verkörpert einen Riesen und tritt gegen einen oder mehrere PC-Krieger in rasanten, explosiven Kämpfen an!“ [3]

Abbildung 2.2: VR-Riese und PC-Charakter in dem Spiel „Davigo“[3]

2.2.3 “Carly and the Reaperman“

„Schlüpfen Sie in die Rolle der kürzlich verstorbenen Carly und rennen und springen Sie sich Ihren Weg durch unglaubliche Jump’n’Run-Herausforderungen - oder schlüpfen Sie in die Rolle des Reaperman und verändern Sie die Umgebung um Sie herum in Echtzeit, um Carly neue Orte zum Erkunden zu geben, gefährliche Hindernisse zu umgehen und ihr zu helfen, das Unmögliche zu erreichen.“ [4]

2.3 Erwartete Probleme

Da asymmetrische VR-Spiele kein sehr großes Genre sind, wird man zur Entwicklung nicht viele Informationen aus dem Internet oder Büchern bekommen. Somit wird es bei der Entwicklung einige Schwierigkeiten und Probleme geben, bei denen die Lösung selbst herausgefunden werden muss. Wichtig ist, dass man diese Probleme in verschiedene Bereiche unterteilt, um diese effizienter abzuarbeiten. Es ist wichtig zu wissen, dass von nun an, um Verwirrung zu vermeiden, zwei verschiedene Begriffe verwendet werden. Wenn von einem PC-Spielenden die Rede ist, wird von dem Spielenden berichtet, der das Spiel mit Controller oder Tastatur und Maus erlebt. Der VR-Spielende ist jener Spielende, der die VR-Brille auf dem Kopf trägt und damit das Spiel spielt. [5] [6]

2.3.1 Motion-Sickness und VR-Krankheit

Grundsätzlich steckt hinter dem Begriff „Motion-Sickness“ die Reisekrankheit. Die Reisekrankheit hat fast jeder Mensch schon mal gespürt. Auslöser ist, dass Information von zwei oder mehreren Sinnen nicht miteinander übereinstimmen. Sollten die Augen eines Menschen die Information übermitteln, dass man sich nicht bewegt und der Gleichgewichtssinn nimmt trotzdem eine Bewegung wahr, kann somit passieren, dass man sich unwohl oder schlecht fühlt. Die VR-Krankheit ist jedoch eine Unterform der Reisekrankheit (so wie die Seekrankheit). Die VR-Krankheit (engl. Virtual-Reality-Sickness) ist eine spezielle Form von Übelkeit, die auftreten kann, wenn man sich in einer virtuellen Umgebung befindet. Die Hauptsymptome sind Kopfschmerzen, Apathie, Müdigkeit und Unwohlsein. Bewegungsinstabilität und Stolpern sind auch mögliche Symptome. In weniger häufigen Fällen wurde Übelkeit und Erbrechen beobachtet. Die Symptome der VR-Krankheit ähneln denen der Seekrankheit sehr, jedoch gibt es einen entscheidenden Unterschied. Bei der Seekrankheit ist der Mensch physisch in Bewegung und seine Sinne geben zusätzlich zu visuellen Informationen auch noch Informationen über seine Bewegung. Im Gegensatz dazu tritt die VR-Krankheit ohne von außen induzierter Bewegung auf. Das heißt, dass die Übelkeit nur durch visuelle und akustische Änderung auftritt. Die VR-Krankheit entsteht dadurch, dass die körperliche Selbstwahrnehmung (Propriozeption) von der Wahrnehmung des visuellen Cortex abweicht. [5] [6]

Mögliche Auslöser in VR:

- Abrupte Bewegungen,
- Schnelle Bewegungen im Spiel,
- Bewegung und Drehung des Spielenden durch einen Analog-Stick.

- Schnelle Positionsänderung in der Spielwelt.
- Kamerafahrten (z.B. Achterbahnen)

2.3.2 Was ist die Frame-Rate?

Die Frame-Rate ist die Bildrate (engl. Frames per Second). Die wissenschaftliche Definition der Bildrate sagt, dass es sich um die Anzahl der Bilder handelt, die in einer einzigen Sekunde gezeigt werden. Dabei ist egal, in welchem Medium. 120 FPS bedeuten also, dass man in jeder Sekunde 120 Bilder sieht. Dadurch, dass in einem kurzen Zeitraum so viele Bilder gezeigt werden, sorgt es für eine flüssige Bewegung. (vergleiche [7])

2.3.3 Wenige Bilder pro Sekunde

Im Optimalfall lautet die Regel für die Menge an FPS: Je höher, desto besser. Alles unter 60 FPS führt wahrscheinlich zu Übelkeit, Desorientierung und allgemeinem Unbehagen, dass zu Motion Sickness führen kann. 120 FPS pro Auge sind am besten für ein gutes VR-Erlebnis geeignet, und alles, was darüber liegt, ist umso besser.

(vergleiche [7])

2.3.4 Platz und Bewegung

Je nach VR-Brille kann der Platzverbrauch des HMDs variieren. Manche Brillen brauchen zusätzlich zu den Controllern und des Head Mounted Displays auch noch "Lighthouses"(Stationäre Sensoren/Kameras, die die Bewegung des Spielenden nachvollziehen). Dieses können zusätzlich etwas Platz verbrauchen und den Spielraum etwas einschränken. Schnelle und hektische Bewegungen, die durch ein gutes Spielerlebnis entstehen, können so oftmals darin enden, einen Schrank, Schreibtisch oder Bildschirm zu beschädigen. Somit ist es also auch wichtig, ein Spiel möglich platzsparend zu bauen, um Spielenden mit wenig Platz auch eine Möglichkeit zu bieten, das Spiel zu spielen.

2.3.5 Interaktionen

Gut ausgearbeitete Interaktionen sind eine wichtige Grundlage für ein VR-Spiel. Flüssige und interessante Interaktion mit der Spielwelt und den Objekten darin kann ein Spiel um einen wesentlichen Bestandteil verbessern und ansprechender machen. Somit erhöhen bessere und auch realistischere Interaktionen die Immersion des Spielenden.

Kapitel 3

Mögliche Performance-Optimierungen

3.1 Level Of Detail

Der Begriff "LOD"(Level of Detail) bezieht sich auf die Komplexität der 3D-Darstellung eines 3D-Models. Das LOD kann mit der Distanz vom Betrachter oder nach Wichtigkeit des Objektes verringert oder erhöht werden. LOD kann die Geschwindigkeit und Effizienz des Renderings erhöhen, indem die Arbeitslast in der Grafikpipeline verringert wird. Diese Performance-Verbesserung wird dadurch erhalten, dass beispielsweise weniger Vertices gerendert werden müssen. Die verringerte visuelle Qualität ist oft nicht bemerkbar, da Objekte sich dann in erhöhter Distanz befinden oder sich schnell bewegen. LOD wird meistens nur auf Geometriedetails angewendet, jedoch kann dieses Konzept auch anders angewendet werden. Theoretisch kann man auch Shader und Pixelkomplexität basierend auf diesen Parametern abändern. Das sogenannte "Mip-Mapping", das seit einigen Jahren existiert, kann ebenfalls eine höhere Rendering-Qualität bieten. [8]

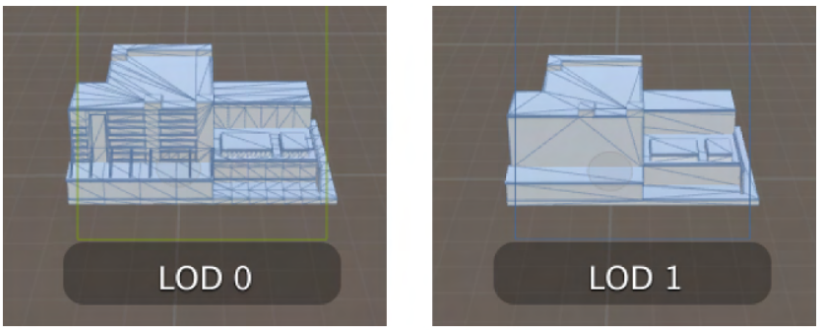


Abbildung 3.1: Ein Mesh mit zwei verschiedenen Detailgraden[8]

3.2 Culling

Beim 3D-Rendering beschreibt der Begriff "Culling"das frühzeitige Aussortieren von Objekten jeglicher Art, die nicht zum endgültigen Bild beitragen. Es gibt viele Techniken, die Objekte in verschiedenen Bereichen der Rendering-Pipeline auszusortieren. Einige dieser Techniken werden vollständig in Software auf der CPU ausgeführt, andere werden von der Hardware (GPU) unterstützt und wieder andere sind in die Grafikkarte integriert. Es ist hilfreich, alle diese Techniken zu verstehen, um eine gute Leistung zu erzielen. Um viel Rechenkraft zu sparen, ist es wichtig, frühzeitig und in größerem Umfang zu Cullen. Wichtig ist auch, dass das Culling an sich selbst nicht zu viel Leistung und Speicherplatz kostet. [9] Normalerweise erfolgt das Culling der Engine auf der Draw Call Ebene. Beim Culling werden nicht die einzelnen Triangles gecullt, sondern ganze Objekte. Das bedeutet, dass es sinnvoll sein kann, große Objekte in mehrere Teile aufzuteilen, sodass nicht alle Teile zusammen gerendert werden.[9] Was vom Culling ausgeblendet wird, muss auch nicht von der GPU berechnet werden. [9]

3.2.1 Beispiel

Ein Objekt, bei dem Culling angewandt werden könnte, ist beispielsweise ein Haus, das einen Innenraum besitzt. Dieses Haus besteht aus 5 Millionen Triangles. Sieht man das Gebäude, dann sieht man also 5 Millionen Triangles. Sieht man es nicht, rendert man 5 Millionen Triangles weniger. Nun könnte man optimieren.[9] Der wichtigste Schritt bei so einem Haus wäre, den Innenraum von der äußeren Fassade zu trennen. So kann man den Innenraum Cullen, falls man sich außerhalb des Gebäudes befindet und spart beispielsweise die Hälfte der Triangles ein. Umgekehrt kann man die Fassade Cullen, wenn man sich innerhalb des Gebäudes aufhält.[9] Zusätzlich kann man jetzt noch Level of Detail für die äußere Fassade verwenden, um Triangles zu sparen, während man nicht in unmittelbarer Nähe des Gebäudes befindet. Sollte man ein detailliertes Gebäude implementieren wollen, ist es oft klug, dieses Gebäude in mehrere Teile zu teilen und auf diesen zusätzlich LODs zu verwenden. Culling ist also wesentlich mehr als nur eine Funktion und es muss vorher überlegt werden, wo und wie es angewandt wird. [9]

3.2.2 Frustum-Culling

"Frustrum-Cullingïst eine wichtige Technik, die in so gut wie jeder ernst zu nehmenden Engine vorhanden ist. Die Grundlage des Frustrum-Cullings ist, Objekte, die sich nicht in der Sehpyramide befinden (also nicht gesehen werden und keinen Einfluss auf das endgültige Bild haben), nicht zu rendern und somit Rechenleistung zu sparen. [9]

Öcclusion-Culling"überprüft, ob Objekte oder auch Polygone von ande-

ren verdeckt werden und rendert diese dann nicht. Diese Technik ist besonders bei Szenen mit hoher Tiefenkomplexität äußerst wichtig und kann zu erheblichen Performance-Einsparungen führen. [9]

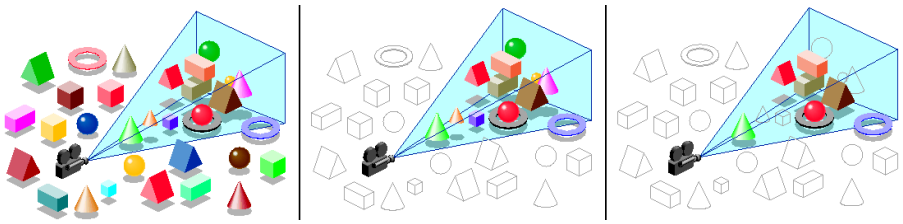


Abbildung 3.2: Sehpyramide ohne Culling, mit Frustrum Culling und mit Frustrum und Occlusion Culling[10]

3.2.3 Backface-Culling

"Backface-Cullingïst mittlerweile ein Standardfeature, dass die Ausrichtung der Polygone eines Meshes überprüft und damit festlegt, ob diese gerendert werden sollen. [9]

3.3 Draw-Calls

In einem Spiel werden unterschiedliche Aufgaben entweder von GPU oder CPU berechnet. Aufgaben, die die CPU und die GPU weitergibt, nennt man "Draw-Calls". Man kann sich Draw-Calls bildhaft als Briefe vorstellen, die die CPU und die GPU weitersendet. Umso mehr Draw-Calls geschickt werden, umso mehr Draw-Calls muss die GPU verarbeiten. Das Empfangen eines Draw-Calls dauert in der Regel länger als diesen Call zu verarbeiten. Grundsätzlich (ohne andere Optimierungen) wird für jedes einzelne Mesh ein Draw-Call versendet. Somit ist es vorteilhaft, wenn möglich, viele kleine Objekte in ein großes Mesh zu kombinieren. So gibt es weniger Draw-Calls, jedoch werden die Draw-Calls größer. Man muss also eine gute Balance zwischen Menge und Größe finden. Grundsätzlich kann man jedoch grob behaupten, dass man möglichst wenige Draw-Calls senden sollte. Draw-Calls werden aber nicht nur von Meshes verursacht, sondern können beispielsweise auch aus Schattenberechnungen und Reflektionen entstehen. [11]

3.3.1 Dynamic-Draw-Call-Batching

Die Unity-Engine stapelt Draw-Calls von mehreren Game-Objects, die das gleiche Material verwenden. Somit wird die Anzahl der Draw-Calls reduziert und die Last auf der Grafikkarte verringert. Die besten Ergebnisse können erzielt werden, indem dasselbe Material auf möglichst viele Objekte angewandt wird. Um diese Technik anzuwenden, muss man jedoch

auf ein paar Regulierungen achten. Mesh-Renderer, Trail-Renderer, Line-Renderer, Particle-Systems und Sprite-Renderer unterstützen Draw-Call-Batching. Andere Rendering Komponenten unterstützen diese Technik nicht. Abgesehen davon muss die Rendering-Komponente dieselbe sein. Das heißt, dass beispielsweise.Mesh-Renderer und Trail-Renderer nicht gebatcht werden. Sollte man mehrere Materialien verwenden, ist es auch möglich, die gleiche Textur zu einzusetzen und damit Performance zu sparen. Dieser Vorgang heißt "Texture-Atlasing". [11]

3.3.2 Static-Draw-Call-Batching

Die Unity-Engine kann Static-Batching während der Build und Runtime durchführen. Generell sollte jedes Objekt, das in einer Szene existiert, durch den Editor in der Build Time gebatcht werden. Sollte ein Objekt erst später erzeugt werden, kann dieses jedoch auch mithilfe der Runtime API gebatcht werden. Wenn die Runtime API verwendet wird, kann das Rootobjekt auch später noch skaliert, rotiert und transformiert werden. Die einzelnen Meshes, aus denen dieser Batch besteht, jedoch nicht. Um Static-Batching verwenden zu können, muss die Gruppe an Game-Objects einige Kriterien erfüllen: [11]

„The GameObject is active. The GameObject has a Mesh Filter component, and that component is enabled. The Mesh Filter component has a reference to a Mesh. The mesh is read/write enabled. The mesh has a vertex count greater than 0. The mesh has not already been combined with another Mesh. The GameObject has a Mesh Renderer component, and that component is enabled. The Mesh Renderer component does not use any Material with a shader that has the DisableBatching tag set to true. Meshes you want to batch together use the same vertex attributes. For example Unity can batch meshes that use vertex position, vertex normal, and one UV with one another, but not with meshes that use vertex position, vertex normal, UV0, UV1, and vertex tangent.“ (siehe [11])

3.4 Object-Pooling

Wenn Game-Objects schnell erstellt und gelöscht werden müssen, ist Object-Pooling eine gute Technik, um Projekte zu optimieren und die CPU-Last zu verringern. Object-Pooling ist ein Design-Pattern, bei dem die benötigten Objekte bereits vor dem Start des Spiels erstellt und in ein „Zwischenlager“ geschoben werden. Bei Anwendung werden diese dann aus dem Lager geholt und anstatt diese Objekte nach Gebrauch zu zerstören, werden diese dann wieder ins Lager verschoben. Bei manchen Anwendungsfällen erhöht Object-Pooling die Performance erheblich. Ein gutes Beispiel wären unter anderem Projektile in einem Top-Down-Shooter. [12]

3.5 Coroutines vs Update

In Unity werden sowohl Coroutines als auch die Update-Methode verwendet, um den Zustand eines Spiels oder einer Anwendung im Laufe der Zeit zu aktualisieren. Sie funktionieren jedoch auf leicht unterschiedliche Weise, was sich auf ihre Leistungsmerkmale auswirken kann. Die Update-Methode ist eine eingebaute Unity-Funktion, die einmal pro Frame aufgerufen wird und typischerweise zur Aktualisierung der Positionen, Geschwindigkeiten und anderer Eigenschaften von Objekten in der Szene verwendet wird. Diese Methode wird von Unity automatisch aufgerufen. Coroutines sind eine Möglichkeit, benutzerdefinierte Iterationen zu erstellen, die gleichzeitig mit der Update-Methode ausgeführt werden. Eine Coroutine ist im Wesentlichen eine Funktion, die Angehalten und zu einem späteren Zeitpunkt wieder aufgenommen werden kann. Coroutines können verwendet werden, um einen Prozess durchzuführen, der über einen bestimmten Zeitraum hinweg erfolgen soll, z. B. das Verschieben eines Objekts an eine neue Position oder das Abspielen einer Animation. Coroutines sind weniger Rechen-effizient als die Update-Methode. Wenn der Zustand eines Objekts sich einmal pro Frame aktualisieren soll, ist es in der Regel am besten, die Update-Methode zu verwenden. Dadurch wird sichergestellt, dass die Änderungen mit dem Rest des Spiels synchronisiert sind, und es benötigt weniger Rechenleistung. Wenn man eine Funktion immer wieder pausieren muss und diese vielleicht sogar in anderen Abständen als jeden Frame aufrufen will, dann ist eine Coroutine die bessere Wahl. [13]

3.6 Statische -und Dynamische Collider

Ein Objekt, auf dem sich ein Collider befindet, kann je nach den Einstellungen, die in dem zugehörigen Rigidbody gesetzt sind, unterschiedlich mit anderen Collidern interagieren. Die drei wichtigsten Einstellungsmöglichkeiten sind: Static-Collider(Kein Rigidbody), Rigidbody-Collider, Kinematic-Rigidbody-Collider. [14]

3.6.1 Static-Collider

Ein statischer Collider ist ein Collider auf dem sich ein Collider aber kein Rigidbody befindet. Statische Collider werden normalerweise bei Levelgeometrie, die sich nicht bewegt, verwendet. Statische Collider, können von anderen Collider nicht bewegt werden. In bestimmten Fällen optimiert die Physics-Engine statische Collider, die sich nicht bewegen. Statische Collider können während der Programmlaufzeit an und aus geschalten und bewegt werden, ohne dass die Rechenzeit der Physics-Engine wesentlich erhöht wird. [14]

3.6.2 Rigidbody-Collider

Ein Rigidbody-Collider ist ein Game-Object mit einem Collider und einem normalen nicht-kinematischem Rigidbody. Rigidbody-Collider sind vollständig von der Physics-Engine simuliert und können auf Kollisionen und Kräfte-Einwirkungen reagieren. Sie können mit anderen Objekten (auch statischen Collider) kollidieren und sind die am meisten verwendeten Collider in einem Spiel, die Physics verwenden. [14]

3.6.3 Kinematic-Rigidbody-Collider

Ein Kinematic-Rigidbody-Collider ist ein Game-Object auf dem sich ein Collider und ein kinematischer Rigidbody befinden. Ein kinematischer Rigidbody ist ein Rigidbody, bei dem die ÏsKinematic-Propertyäktiviert ist. Ein kinematischer Rigidbody kann zwar anhand seines Transforms bewegt werden, aber er kann nicht wie der nicht kinematische Rigidbody, auf Kollisionen und Kräfte-Einwirkungen reagieren. Ein Kinematischer Rigidbody-Collider wird dann verwendet, wenn ein Objekt manchmal mit einem Collider bewegt wird, es aber sonst wie ein statischer Collider agiert.[14] Ein Beispiel dafür wäre eine Schiebetür, die normalerweise ein unbewegliches Hindernis ist, sich aber wenn nötig öffnen kann. Im Gegensatz zu einem statischen Collider kann ein bewegender Kinematic-Rigidbody Reibung an andere Objekte weitergeben und „weckt“ andere Rigidbodies bei Kontakt auf. Auch wenn dieser sich nicht bewegt, hat dieser ein anderes Verhalten als ein statischer Collider. Beispielsweise kann ein Collider, der als Trigger gesetzt ist, nur Trigger Events empfangen, wenn sich ein Rigidbody auf dem Game-Object befindet. Sollte gewollt sein, dass dieser Trigger nicht von Physik beeinflusst werden kann, dann kann die ÏsKinematic-Propertyäuf dem zugehörigen Rigidbody aktiviert werden. Ein Rigidbody kann immer zwischen normal und kinematisch mit Hilfe der ÏsKinematic-Property", gewechselt werden. Ein Rigidbody kann immer zwischen normal und kinematisch mithilfe der ÏsKinematic-Property"gewechselt werden. Ein gängiges Beispiel dafür wäre der „Ragdoll“ Effekt, bei dem sich ein Charakter normalerweise mit Animationen bewegt, aber physikalisch geworfen werden kann. Beispielsweise könnte eine Explosion oder starke Kollision diesen Charakter schleudern. Es kann jedem Körperteil des Charakters ein Rigidbody zugewiesen werden, der auf ÏsKinematic"gesetzt ist. Die Körperteile bewegen sich dann normal, mit der Animation, bis ÏsKinematicäusgeschalten wird und der Charakter sich wie ein Physikobjekt verhält. Jetzt könnte eine Explosion oder Kollision diesen Charakter werfen, um eine glaubwürdige Reaktion auf diese Explosion zu bieten.

[14]

Kapitel 4

Umsetzung der PC- und VR-Welt

4.1 Levelaufbau

Jedes Level in *Tricks ´n´ Treats* ist ähnlich aufgebaut. In der Mitte befindet sich der Berg, auf dem die Snowboardenden fahren, und rund herum befindet sich der Raum des VR-Spielenden. Die PC-Spielenden starten oben auf dem Berggipfel und fahren den Hang bis zum Ende der Piste hinab. Der VR-Spielende kann mit einem Joystick auf seinem Controller den Berg rotieren, um sich besser auf der Strecke zu orientieren und die Spielenden in Sicht zu behalten. Währenddessen kann er hinter sich bestimmte Zauberkugeln aufheben und auf den PC-Spielenden anwenden.

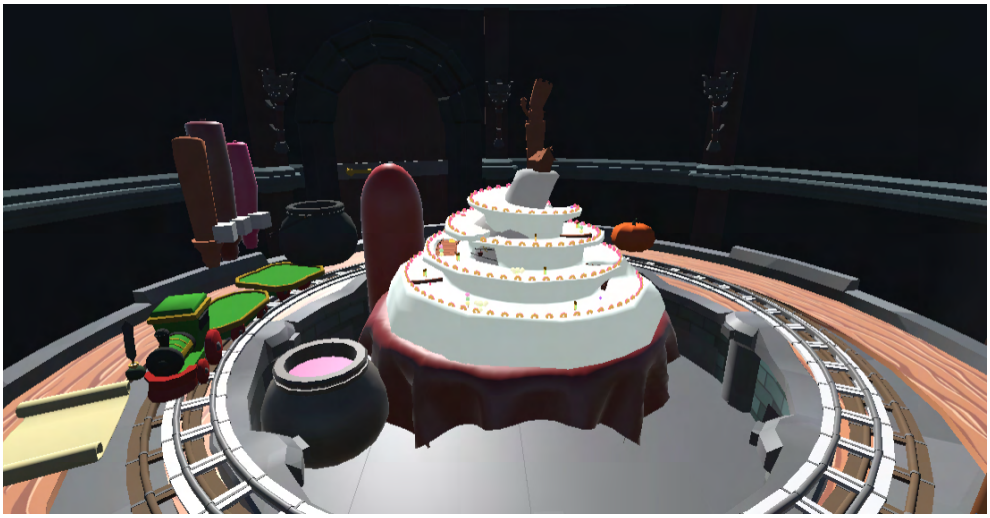


Abbildung 4.1: Editor View eines unfertigen Levels, Die rote Kapsel stellt den VR-Charakter dar

4.2 VR-Charakter

Der VR-Charakter besteht aus mehreren Systemen, die zusammenarbeiten. In *Tricks ´n´ Treats* sind die Kamera und Hände mithilfe der Steam-VR Integration für Unity gemacht. Der VR-Charakter verwendet die physikalischen Hände und die „Player“ Komponente, die beide von Steam-VR bereitgestellt werden. Ein großes Problem unseres Spiels ist, dass der VR-Charakter, der physikalische Hände besitzt, nicht richtig skaliert werden kann. Sobald dieser skaliert wird, funktionieren die Hand-Collider nicht mehr planmäßig. Die Lösung dieses Problems ist in 4.4 erklärt.

4.2.1 Zauber -und Interaktionssystem

Für die Basis des Interaktionssystems in *Tricks ´n´ Treats*, dient das System, das Steam-VR für Unity bereitstellt. Auf jedem Objekt, mit dem der VR-Spielende interagieren kann, befindet sich das „Interactable“ Script, dass von Steam-VR bereitgestellt ist. Unser eigenes System für die Zauberkugeln hakt sich in dieses Script ein und kann somit bestimmte Eigenschaften leichter implementieren. Somit wurde eine Hervorhebung für Objekte geschaffen, mit denen interagiert werden kann und gleichzeitig wird auf die Eingabe des VR-Controllers reagiert. Wenn eine Zauberkugel mit einem Objekt kollidiert (während man sie in der Hand hält), wird dann mit dem von Steam vorgegebenen, „Velocity Estimator“ die Aufprallgeschwindigkeit ausgerechnet und je nach Geschwindigkeit ein Zauberspruch gewirkt.

4.2.2 VR-Umgebung

Die Form der VR-Umgebung basiert auf einem Kreis, bei dem sich der Berg in der Mitte befindet und der Spielbereich des VR-Spielenden auf einem Ring um den Berg.

4.3 PC-Charakter

4.3.1 Charakter-Größe

Die PC-Charaktere sind sehr klein. Das kommt daher, dass der VR-Charakter nicht skaliert werden kann und daher die PC-Charaktere runter skaliert werden müssen. Das Iterieren der Werte für den Character-Controller ist durch die geringe Größe etwas komplexer und ungewohnt. Dadurch, dass die PC-Charaktere so viel kleiner sind, muss auch die Schwerkraft anders eingestellt sein. Diese haben eine viel geringere Schwerkraft als die physikalischen Objekte, die sich im VR-Bereich befinden.

4.3.2 Fähigkeiten

Der PC-Character-Controller kann lenken, springen und Tricks machen. Außerdem muss dieser auf bestimmte Zauber des VR-Charakters reagieren. Somit kann der PC-Charakter beispielsweise künstlich in eine Richtung geschleudert werden, wobei er auch eine Rotation bekommt. Die PC-Spielenden können dann in der Luft den jeweiligen Charakter austarieren und bei richtiger Ausführung wieder sicher landen. Sollte der Charakter auf dem Kopf oder in einem falschen Winkel landen, stürzt dieser und wird zu einem Ragdoll (=physikalischer Körper, bei dem jedes Körperteil eigene Kollisionen hat und somit einen realistischen Fall simuliert). Der Charakter kann zusätzlich, um besser ausweichen zu können, vorwärts und rückwärts Saltos, mit denen er schnell seine Position ändern kann.

4.4 Lösung der erwarteten Probleme

4.4.1 Wenige Bilder pro Sekunde

Um möglichst viele "Fps"(Bilder pro Sekunde/Frames per second) zu erreichen und das Spiel auch auf älteren Systemen spielbar zu machen, muss es gut optimiert sein. Die Möglichkeiten zur Rechenoptimierung aus 3, die verwendet wurden und die, die nicht implementierbar waren, sind anbei zu sehen.

4.4.2 Level Of Detail

Die Unity-Engine unterstützt "Level of Detail per Camera". Das heißt, dass jede Kamera selbst berechnet, welche Meshqualität gerendert werden soll. In *Tricks ´n´ Treats* wird diese Technik nicht verwendet, weil durch andere Optimierungen genug Rechenleistung gespart werden konnte. Sollte sich der Spielumfang in Zukunft ändern, wird es essenziellsein, mehr LOD zu implementieren.

4.4.3 Culling

Dadurch, dass die PC -und VR-Kamera gleichzeitig Rendern, kann Culling nur begrenzt angewendet werden. Occlusion-Culling könnte durch eigenen Code, der anhand des View-Frustrums die Objekte prüft und dementsprechend cullt, implementiert werden. Eine Frage, die vorab gestellt werden muss, ist jedoch: Entsteht dadurch eine nennenswerte Verbesserung? Man kann nur testen, ob der Code nicht mehr Rechenleistung braucht als der Renderingprozess der zusätzlichen Objekte. Da durch andere Maßnahmen genug optimiert wurde, konnte die Zeit, um in diesem Bereich zu forschen, anders verwendet werden.

4.4.4 Collider

In *Tricks ´n´ Treats* bestehen alle Objekte, mit denen kollidiert werden kann aus simplen Collider Formen und komplexere Strukturen werden aus mehreren primitiven gebaut. Für die Effizienz der Collider gilt Sphere>Capsule>Box>Mesh. Das heißt, dass ein Sphere-Collider am effizientesten ist und ein Mesh-Collider am ineffizientesten.

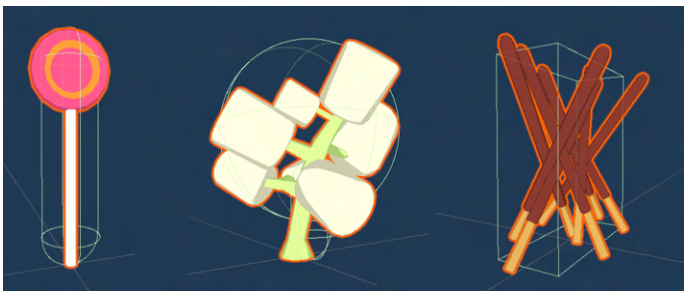


Abbildung 4.2: Korrekte Collider

4.4.5 Object-Pooling

In *Tricks ´n´ Treats* gibt es keine Objekte, die gepoolt werden könnten, um Rechenleistung zu sparen. Somit gibt es im gesamten Projekt kein Pooling System.

4.4.6 Drawcall-Batching

In *Tricks ´n´ Treats* wird sowohl Static als auch Dynamic-Batching angewendet. Um dynamisch zu batchen werden Meshes mit weniger Vertices verwendet, wobei alle dasselbe Material nutzen. Für die Textur des Materials wird ein Textur-Atlas verwendet, der alle Farben aller Meshes beinhaltet. Dies gilt für alle Meshes der Umgebung. Da die Spielenden wegen einiger Rahmenbedingungen des Dynamic-Batching nicht gebatcht werden können, verwenden diese einen anderen Atlas und Material. SStaticïst alles, was sich aus der Sicht der PC-Spielenden nicht bewegt. Das heißt beispielsweise der Berg und die Objekte auf diesem. Hindernisse, die zerbrechen oder sich bewegen können jedoch nicht SStaticßein.

4.4.7 Bewegungskrankheit und Rotationsproblem

Um die Bewegungskrankheit (Motion-Sickness) zu vermeiden, wurden die in 2.3.1 angebrachten Ursachen dafür beachtet und zugehörige Systeme passend dazu entworfen und implementiert. Wie in 4.2 angeführt wurde, kann man den VR-Spielenden nicht skalieren und sondern ihn nur drehen. Die Strecke wiederum kann skaliert, aber nicht gedreht werden. Das heißt, dass

also der VR-Spielenden in einem Ring um den Berg gedreht werden muss, um den Spielenden folgen zu können. Einige Menschen werden bei einer derartigen Drehung jedoch seekrank. Damit dem VR-Spielenden nicht übel oder schwindelig wird, muss diesem das Gefühl gegeben werden, dass er den Berg rotiert und nicht er sich um den Berg. Um das zu umgehen, wird der gesamte Raum des VR-Spielenden und die sich darin befindenden Lichter zusammen mit dem VR-Spielenden gedreht. Somit wirkt es für den VR-Spielenden so, als würde sich eigentlich nur der Berg drehen, da für ihn alles andere statisch wirkt.

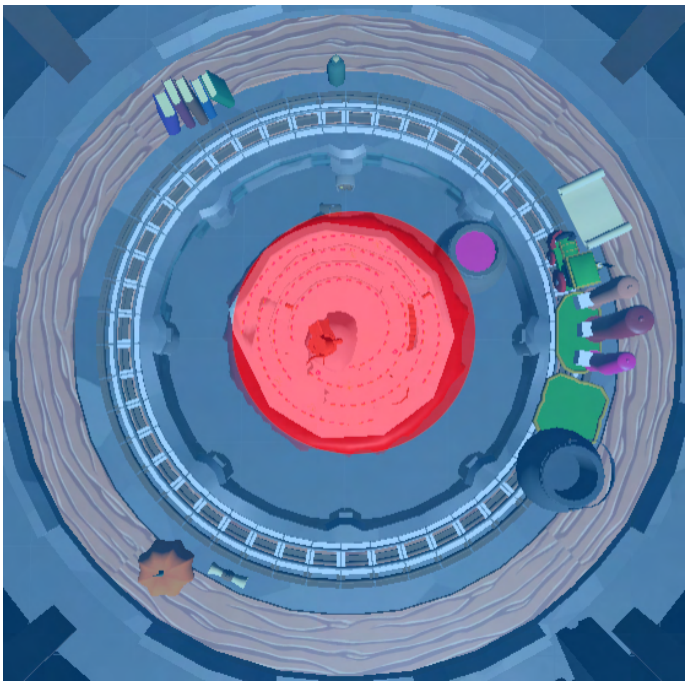


Abbildung 4.3: Top Down Layout des VR-Raumes mit dem Berg in der Mitte, blaue Zone stellt rotierbaren Bereich dar

4.4.8 Platz und Bewegung

Um möglichst vielen Menschen das Spielen von *Tricks ´n´ Treats* zu ermöglichen, ist der Spielbereich so klein wie möglich, ohne ihn einengend zu gestalten. Das heißt, dass alles Wichtige in Griffreichweite ist und stationär erreicht werden kann. Personen, die mehr Platz zur Verfügung haben, können diesen jedoch auch nutzen. Abgesehen davon muss der VR-Spielende sich nicht im Ring um den Berg bewegen, da man den Berg augenscheinlich drehen kann.

4.4.9 Interaktionen

Die Interaktionen des VR-Spielenden müssen sich real anfühlen, um die Immersion nicht zu brechen. Deswegen werden in *Tricks ´n´ Treats* physikalische Hände verwendet, die nicht durch Objekte durch transformieren können. Außerdem vibriert die jeweilige Hand etwas, sobald man versucht, durch ein Objekt zu greifen. Damit wird das Gefühl der Berührung ersetzt, dass mit den meisten VR-Controllern nicht umsetzbar ist.

Interaktion mit Knöpfen und Items

Der VR-Spielende kann Zauberkugeln aufheben. Wenn diese aufgehoben werden, nimmt die virtuelle Hand eine greifende Pose ein. Um die Zauberkugel nun gegen die PC-Spielenden einzusetzen, kann der VR-Spielende diese Kugel auf dem Berg platzieren, indem er diese auf die Oberfläche schlagt. Man kann die Interaktion damit vergleichen, ein Ei auf einen Tisch zu schlagen. Um mit Knöpfen zu interagieren, gibt es zwei Methoden. Die erste Methode ist, den Knopf so zu machen, dass man nur mit der Hand in die Nähe einer unsichtbaren Zone kommen muss, um den Knopf hineinzudrücken. Die zweite und auch immersivere Methode ist, einen physikalischen Knopf zu implementieren, der vom Gefühl näher an einen realen Knopfdruck herankommt.

Kapitel 5

Fazit

Die Entwicklung von asymmetrischen VR-Spielen birgt mehr Schwierigkeiten als ursprünglich erwartet. Wenn man von der Rechenintensivität absieht, zeigen sich einige Probleme erst mitten in der Entwicklungszeit. Mehrere Kameras gleichzeitig rendern zu müssen, senkt nicht nur die Performance, sondern erschwert auch die Möglichkeit, diese zu optimieren. Die verschiedenen Größenverhältnisse des VR-Charakters und der Snowboardenden verkomplizieren die Implementierung raumübergreifender Systeme, in der diese miteinander interagieren können. Die größte Schwierigkeit der Entwicklung bestand aus den vielen, sehr unterschiedlichen Systemen, die miteinander arbeiten müssen. Aufgrund der simplen 3D Modelle, die verwendet werden, konnte durch ein paar simplere Optimierungen schon sehr viel Rechenleistung gespart werden. Somit war es nicht notwendig, komplizierte Optimierungen zu Implementieren. Die Herausforderungen, die Asymmetrisches Virtual Reality mit sich gebracht hat, konnten durch Zusammenarbeit gut bewältigt werden und das Arbeiten innerhalb des Teams hat bis auf ein paar Ausnahmen sehr gut funktioniert. Dadurch, dass die Artists nichts über den Code und das Game-Design des Spiels wissen mussten, wurden Informationen über Änderungen nur zwischen zwei Personen weitergegeben, was den Prozess erheblich erleichtert hat. Wenn *Tricks ´n´ Treats* in Zukunft weiterentwickelt werden würde, müssten einige Änderungen durchgeführt werden. Um das Spiel grafisch zu skalieren, müsste LOD implementiert werden und einige Systeme, welche das Local-Multiplayer handhaben, müssten neu oder umgeschrieben werden, um zukünftig fehlerfrei zu bleiben. Sollten zukünftige Playtests zeigen, dass das Spiel unter den Testenden gut ankommt, ist es erstrebenswert, diese komplexen Probleme zu beheben und weitere Arbeitszeit in das Projekt zu investieren.

Design eines asymmetrischen Local-Multiplayer-Party-Games unter Verwendung des MDA-Frameworks

Felix Kaspar

D I P L O M A R B E I T

eingereicht am
Ausbildungsschwerpunkt
Mediendesign - Gamedesign

betreut durch
René Ksuz, BSc MA

im März 2023

Kapitel 6

Begriffsdefinitionen

Bei „Tricks 'n' Treats“ handelt es sich um ein asymmetrisches Couch-Party VR-Spiel.

In diesem Kapitel werden die in „Tricks 'n' Treats“ verwendeten Technologien erklärt und die grundlegenden Konzepte des Game-Designs, die verwendet werden, um die Erlebnisse der Spielenden und die Interaktionen zwischen ihnen zu beeinflussen.

6.1 VR trifft Couchparty-coop

6.1.1 Was ist Virtual Reality?

Virtual-Reality bezeichnet Bilder und Töne, die von einem Computer erzeugt werden und dem Benutzer, der mithilfe von Sensoren mit ihnen interagieren kann, fast real erscheinen[15]. Sie ist trotz anderer Anwendungsbereiche dank der Immersion und Interaktionsmöglichkeiten, die sie bietet, besonders in der Gaming-Industrie relevant geworden[16]. Spielenden können neue Erlebnisse geboten werden, welche wiederum eigene Game-Design-Fragen aufwerfen, die es zu beantworten gilt.

VR bringt neben der Interaktivität noch einen weiteren gewaltigen Vorteil: Intuition. Greifen, Umschauen, Bewegen, alles funktioniert, wie man es erwartet. Gaming-Neulinge haben häufig Probleme, die richtigen Tasten auf der Tastatur zu drücken und auch klassische Gamepads sind nicht optimal. In VR gibt es zum einen weniger Knöpfe an den Controllern, die gedrückt werden können und zum anderen können alltägliche Interaktionen wie Umschauen / Orientieren und das in die Hand nehmen von Objekten, in VR unterbewusster durchgeführt werden und sind daher für Anfangende weniger frustrierend. Spielende verstehen in VR schneller, wie das Spiel gesteuert wird und können sich mehr Gedanken darüber machen, was das Ziel des Spiels ist[17]. Insbesondere für Party-Spiele (siehe 6.1.2) ist das schelle Verstehen von der Steuerung und der intuitiven Ausführung von grundlegenden Aktionen in Stresssituationen sehr wichtig.

Abbildung 6.1: Grab in Portal[18] vs Alyx[19]

6.1.2 Was ist Couchparty?

Couchparty-Spiele, oft auch nur Party-Spiele oder Couch-Games, sind Computerspiele, die eine Gruppe (die sich meist untereinander kennt), gemeinsam spielt. Anders als bei klassischen Online-Multiplayer-Spielen werden diese zusammen in einem Raum gespielt. Sie sind jedoch nicht zu verwechseln mit Local-Multiplayer-Games. LAN-Klassiker wie *Counter Strike*[1] oder *Starcraft*[2] werden meist nur von leidenschaftlichen Gamern gespielt, da sie Vorbereitung, Know-How und viel Ausrüstung benötigen. Das Alleinstellungsmerkmal von Party-Spielen ist, dass diese auf nur einem Bildschirm von einer Couch aus daher der Name gespielt werden können und nur minimale Ausrüstung, meistens mehrere Controller, wobei bei manchen auch schon das Handy als Controller fungieren kann, benötigt wird.

In Couchparty-Spielen müssen Spielende schnell verstehen, worum es geht, daher stützen sich viele Spiele auf „Zufall“. Dadurch können Anfangende schnell mit den anderen mitkommen, auch wenn sie das Spiel noch nie zuvor gespielt haben. Eine weitere Möglichkeit, das Spiel auch für schlechtere Spieler zugänglich machen, wäre die Verwendung von „Virtual Skills“[20, S. 165].

Es gibt grundsätzlich zwei verschiedene Arten, wie Couchparty-Spiele den Bildschirm-Platz nutzen können.

- **Shared-Screen**

  In diesem Fall sehen alle Spielende das Gleiche und müssen sich daher auch in der Spielwelt im gleichen Raum aufhalten. Der Vorteil eines solchen Systems ist, dass keine Informationen mehrfach gezeigt werden

[1] Counter Strike ist eine Reihe von taktischen Multiplayer-Ego-Shootern
[2] Starcraft ist ein Echtzeit-Strategiespiel

müssen und der Platz des Bildschirms vor allem bei kleinen Laptop-Bildschirmen oder sogar Handys besser genutzt wird. Außerdem hat es einen Performance-Vorteil, da nicht mehrere Kameras gerendert werden müssen. Leider eignet sich diese Art nicht für jedes Spiel, da sich die Spielenden nicht zu weit voneinander entfernen können, vor allem 3D-Spiele mit einer freien Kamera sind hier sehr eingeschränkt. Besonders eignet sich Shared-Screen-Play für Arena-Games wie z.B. *Super Smash Bros* oder *Overcooked*.

- **Split-Screen**

  Ein Splitscreen-System bietet hier einen großen Vorteil: Freiheit. Alle Spielenden können sich selbstständig bewegen und umschauen, ohne das Erlebnis der anderen zu beeinflussen. Allerdings wird hierfür ein größerer Bildschirm benötigt, da jedem Spielenden nur ein halber oder nur ein Viertel des Bildschirms zugewiesen wird.

Es gibt auch die Möglichkeit, nahtlos zwischen den beiden Systemen zu wechseln, je nachdem, welches System sich gerade besser eignet, um das gewünschte Spielerlebnis zu erzeugen. Laut einer Reddit-Umfrage präferieren Spielende Shared-Screen-Play[21].

6.1.3 Was sind coop-games?

Coop-Games (= Cooperative-Games/Kooperative-Spiele) sind eine besondere Art von Team-basierten-Spielen die besonders auf Gemeinschaft der Spielenden setzt und diese auch durch Game-Design voraussetzt.

Ein Koalitions- oder Strategiespiel ist kooperativ, wenn die Spielenden verbindliche Vereinbarungen über die Verteilung der Auszahlungen oder die Wahl der Strategien treffen können, auch wenn diese Vereinbarungen nicht durch die Spielregeln spezifiziert oder impliziert sind[22].

Bei klassischen Team-basierten-Spielen ist Kommunikation zwar wichtig, jedoch nicht essenziell. Aufgaben der Spielenden werden in Coop-Games so verteilt, dass ein Ziel nicht ohne die Zusammenarbeit der anderen erreicht werden kann. Es entsteht hierbei, da die Spielenden komplett voneinander abhängig sind, eine interessante Dynamik, die dazu führen kann, dass sich Spannungen oder Konflikte innerhalb des Teams bilden, da eine Person nicht kooperiert oder Aufgaben auf einem anderen Weg lösen will. Kooperation kann auch beziehungsweise soll in den meisten Fällen den Spaßfaktor eines Spieles fördern, mehr dazu in Kapitel 7.1.2.

Gute Beispiele hierfür sind einige Spiele der Lego-Reihe, *Overcooked* und *It Takes Two* welches in Kapitel 8 genauer beschreiben wird.

Wie erzeugt man Kooperation?

Kooperation, auch jene in Videospielen, kann zu tiefen Bindungen zwischen den Spielenden führen. Mit der „Lens of cooperation“[23, S. 311] kann ein Game-Designer besser verstehen, wie Kooperation entsteht und wie man sie fördert. Die wichtigsten Punkte, die Jesse Schell hier anspricht, sind die Notwendigkeit von Kommunikation oder die Tatsache, dass eine Aufgabe nur durch Zusammenarbeit der Spielenden gelöst werden kann. Ein weiterer interessanter Punkt ist Synergie und Anergie, die Veränderung der Stärke einer Gruppe, wenn ein Team zusammenkommt. Synergie beschreibt, dass die Gruppe Stärker als die Summe der Einzelnen ist, Anergie ist das Gegenteil davon. Synergie ist damit in kooperativem Umfeld erstrebenswert.

6.1.4 Was bedeutet Asymmetrie?

Asymmetrie beschreibt im allgemeinen zwei Seiten oder Teile, die nicht die gleiche Größe oder Form haben[15].

In Spielen bezieht sich Asymmetrie auf das (Board-) Design. Die Spielenden haben in einem solchen Spiel ein hoch unterschiedliches Actionset, das bedeutet, sie haben einen anderen Einfluss auf die Spielwelt/andere Mitspieler. In Spielen mit asymmetrischen Wahlmöglichkeiten beginnt jeder Spielende typischerweise mit einer Reihe von Aktionen, die sich stark von denen der anderen unterscheiden, was das „Balancing“ dieser Spiele besonders schwierig macht. Die Herausforderung besteht vor allem darin, den relativen Einfluss einer Aktion auf die Gewinnwahrscheinlichkeit zu bestimmen[24, S. 18].

Kapitel 7

Spaß in Videospielen

7.1 Warum machen Spiele Spaß?

McKee definiert Spaß sehr simpel: „Vergnügen ohne Ziel“. Nach dieser Definition kann jede Tätigkeit, die für Vergnügen ausgeübt wird, Spaß machen[25].

7.1.1 Woher kommt Spaß?

Game-Designer Marc LeBlanc hat acht verschiedene Typen von situationsabhängigem Spaß definiert. Einige davon lassen sich gut auf Situationen des in dieser Arbeit entwickelten Spieles anwenden, um genauer zu verstehen, woher der Spaß in dem Spiel kommen soll: „drama, obstacle, social framework“.

Das Buch „A Theory of Fun“[26] definiert Spaß als das mentale Meistern eines Problems[26, S. 71]. Außerdem meint Raph Koster, dass Spaß aus „richly interpretable situations“[26, S. 40], also Situationen, die den Spielenden mehrere Möglichkeiten geben, ein Problem zu lösen und Kreativität fördern, entsteht.

Spaß kann also nur existieren, wenn Spielende immer neue Probleme lösen können oder bereits gelöste Probleme auf andere Wege („richly interpretable“) lösbar sind. Das Meistern eines Spieles laut seiner Definition kann also nur Spaß machen, wenn sich die Situationen je nach skill-level ändern oder sie auf andere Wege lösbar sind, mehr dazu in Kapitel 7.2.4. Das ständige Ausführen derselben Aktion macht somit keinen Spaß. Raph Koster hat außerdem einige wichtige Grundsteine für Spaß in Spielen gelegt, diese werden im praktischen Teil noch genauer behandelt.

7.1.2 Spaß bei mehrspieler Spielen

Ein weiterer Ursprung von Spaß bzw. Vergnügen lässt sich in sozialen Interaktionen finden[26, S. 72]. Hier unterscheidet man zwischen kooperativen und kompetitiven Interaktionen, beide sind tief in uns verborgene In-

stinkte[20, S. 310]. Einen Unterschied im Spaßfaktor gibt es hier vermutlich nicht[27], und Spiele haben auch die Möglichkeit, beide Verhalten in verschiedenen Situationen zu fördern.

- **Competitive Play**

  Laut Jesse Schell ist ein wichtiger Aspekt von kompetitiven Spaß, Schadenfreude, Freude, die man im Unglück anderer verspürt[20, S. 196]. Um den Wettbewerb zwischen Spielenden oder Teams zu erzeugen, müssen einige Voraussetzungen erfüllt sein. Der wichtigste Aspekt ist, dass sich Spielende untereinander messen. Es muss für erfahrene Spielende möglich sein, mit einer relativen Sicherheit sagen zu können, dass sie nicht von einem Anfangenden geschlagen werden[20, S. 311].

- **Cooperative Play**

  Spiele die Kooperation fördern können in Spielenden positive Gefühle wie soziale Zugehörigkeit, Möglichkeiten zur sozialen Vernetzung und die Förderung der sozialen Integration bieten[28]. Die Analyse deutet auch darauf hin, dass Flow (mehr dazu in Kapitel 7.2.4) in sozialen Spielen auftritt, insbesondere im kooperativen Gameplay[28]. Soziale Kontexte können so die emotionalen Erfahrungen des Spielens verbessern. Wie man Kooperation in Spielen erzeugt wird, wird in Kapitel 9.3.4 erforscht.

Außerdem belegt Quelle [29], dass Spaß, der mit anderen geteilt wird, stärker empfunden wird als einsamer Spaß, das hat vermutlich einen evolutionären Ursprung.

Arena

Party-Spiele finden in der „Arena“ statt[20, S. 65]. Im Vergleich zu anderen „Arena-Spielen“ befinden sich die Spielenden bei Party-Spielen im selben Raum und können sich so auf noch mehr Weisen beeinflussen, die bei klassischen „Arena-Spielen“, z.B. Shooter, nicht möglich sind. Es entsteht ein anderer, persönlicherer Umgang miteinander sowohl innerhalb als auch außerhalb der Spielwelt.

7.2 Wie kann man Spaß „kreieren“?

Wie vorher definiert gibt es verschiedene Arten von Spaß. Das Erlebnis der Spielenden lässt sich hauptsächlich durch die Aktionen beeinflussen, welche den Spielenden zur Verfügung gestellt werden und deren Auswirkungen auf die Spielwelt oder andere Spielende.

7.2.1 MDA-Framework

Das MDA-Framework hilft bei der Analyse eines Spieles und dem iterativen Designprozess. Es wurde von dem Game-Designer Marc LeBlanc mitentwickelt und trennt den „Konsum“ von Spielen in verschiedene Komponenten[30]. Es ist unterteilt in Mechanics, Dynamics und Aesthetics. Eine Ebene beeinflusst immer die darüber Liegenden. Es beginnt mit den Mechanics, diese beeinflussen alle Ebenen darüber.

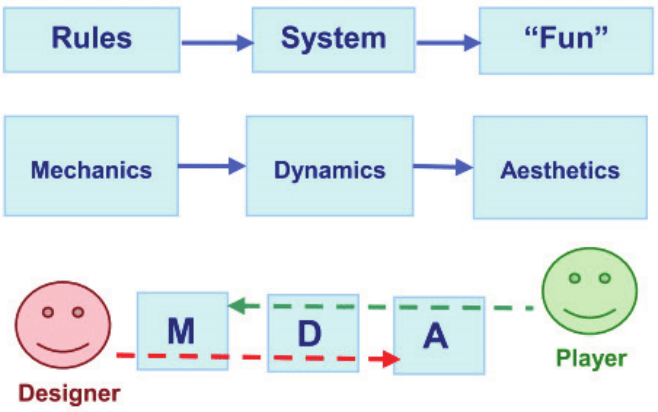


Abbildung 7.1: MDA Framework[30]

- **Mechanics (Regeln)**

  Regeln und Feedback-Loops, die die Spielenden in ihren Aktionen limitieren. Die Mechanics beschreiben das Ziel des Spieles und wie Spielende es erreichen bzw. nicht erreichen können[20, S.96]. Sie beschreiben den Kern des Spieles, selbst wenn man alles andere entfernt bleiben die Mechanics erhalten[20, S.231].

- **Dynamics (System)**

  Das entstehende Verhalten der Spielenden, das aus den Mechanics hervorgeht[30]. Sie beschreiben, wie sich das Spiel bzw. wie sich Systeme und Mechaniken im Laufe der Zeit verändern. Sie bieten dem Spieler ein Gefühl des Fortschritts und der Veränderung, hier spielt auch Feedback eine große Rolle, dazu mehr in Kapitel 7.2.5. Hier kommen die Regeln, die Spielwelt und die Spielenden zusammen, Spielende werden mit der Zeit besser und das Spiel wird schwerer und bietet so eine fesselnde Erfahrung.

- **Aesthetics (Spaß)**

  Die emotionalen Reaktionen der Spielenden. Sobald Spielende zum ersten Mal mit dem Spiel interagieren, löst es Gefühle in ihnen aus. Ziel ist es eben, diese Gefühle zu steuern und das Interesse der Spielenden so aufrecht zu erhalten.

Als Game-Designer muss man das Framework im Überblick behalten. Designer verändern die Mechanics des Spieles, sehen aber gleichzeitig deren Einfluss auf die Aesthetics und müssen diese berücksichtigen, um das Spiel in die richtige Richtung zu lenken. Es ist möglich, den Anfang eines Spieles zu designen, indem man eine interessante Mechanic erstellt und sieht, das Experience daraus entsteht oder indem man Aesthetics festlegt und die Mechaniken des Spieles so anpasst, dass dabei die gewünschte Experience entsteht.

> „Wenn man nicht weiß, was man will, ist es einem vielleicht egal, was man bekommt. Wenn man aber weiß, was man will (...), dann muss man sich überlegen, wie man diese Essential Experience vermittelt.“[20, S.55]

7.2.2 Essential Experience

Oft möchte der Designer eine Experience, ein Erlebnis, das man vorher definiert hat, einfangen. Die „Essential Experience“ versucht, wenn auch auf einem anderen Weg, die reale Experience einzufangen und durch Mechanics zu erzeugen. Das Designen eines Spieles mithilfe der „Essential Experience“ ermöglicht einem, das Spiel von der Experience getrennt zu sehen und so entscheiden zu können, was an dem Spiel noch verändert werden muss bzw. welche Teile des Spieles nicht verändert werden dürfen, um den Grundgedanken zu erhalten[20, S.55].

7.2.3 Core Mechanics

Die Core-Mechanics sind die Grundbausteine/der Anfang des Designprozesses des Spieles. Sie sind jene Mechaniken, die in dem Spieler die erwünschten Gefühle auslösen und ohne welche das Spiel nicht funktionieren könnte. Sie beinhalten die grundlegendsten Dinge wie Interaktionsmöglichkeiten, Fortbewegung und Fähigkeiten[31, S.330].

7.2.4 Pacing, Flow

Der Schwierigkeitsgrad von Spielen muss sich in einem gewissen Bereich aufhalten, der von dem Psychologen Cziksentmihalyi als „Flow Channel“

bezeichnet wird, um die Aufmerksamkeit eines Spielenden zu behalten[20, S.205].

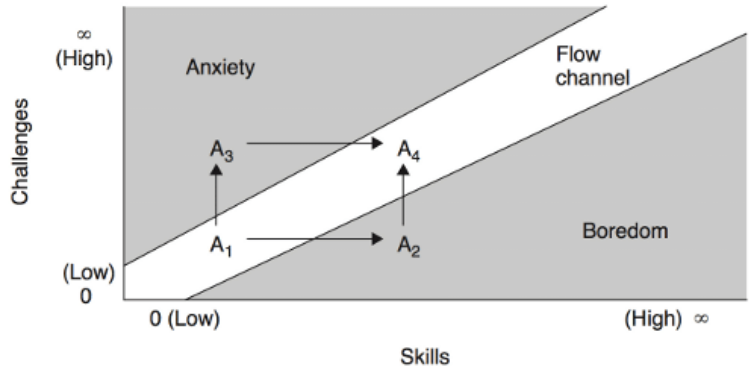


Abbildung 7.2: Flowchannel - The Art of Game Design[20]

Als „Pacing“ beschreibt man die durch die Spielenden gefühlte Intensität und den Rhythmus des Gameplays[32]. Game-Designer haben dabei zum Ziel, die Spielenden in einen Flowstate zu bringen.

Flow ist „ein Gefühl der vollständigen und energiegeladenen Konzentration auf eine Tätigkeit mit einem hohen Maß an Freude und Erfüllung“[20, S.204]. Cziksentmihalyi beschreibt Flow etwas einfacher als einen Zustand, der auftritt, wenn man sich freiwillig an seinem Limit befindet[33]. Laut Jesse Schell müssen es, um Flow zu erzeugen fünf Voraussetzungen gegeben sein[20, S.211]. Die zwei wichtigsten sind klare Ziele, die die Spielenden auch verfolgen und von denen sie nicht abschweifen, und die passende Schwierigkeit und Intensität für die Spielenden. Es ist allerdings zu beachten, dass Flow nicht gleichzusetzen mit Spaß ist. Spaß setzt keinen Flow voraus und es gibt auch Flow, der nicht unbedingt Spaß macht, beispielsweise das Meistern einer Tätigkeit[26, S.75].

7.2.5 Player Action Feedback

Feedback ist ein sehr wichtiger Teil von Spielen. Ohne Feedback wissen Spielende nicht, ob ihre Aktionen richtig sind oder das Spiel sie verarbeitet hat. Visuelles und Auditives Feedback kann die Spielenden über Fortschritt und Fehler informieren. Außerdem hilft es den Spielenden gedanklich vollkommen in die Spielwelt einzutauchen.

Kapitel 8

Stand der Industrie

8.1 Asymmetrical Coop

8.1.1 Keep Talking and Nobody Explodes

In "Keep Talking and Nobody Explodes"übernimmt eine Person das Entschärfen einer Bombe, während die andere eine Anleitung zur Entschärfung navigieren muss. Das Spiel ist zwar kooperativ, jedoch sehr einseitig, da der Bombenentschärfende oft auf den Instruktor warten muss und es, abgesehen von Beschreibungen der Bombe oder der Anleitung nicht viel Kommunikation gibt. Auch kein euphorisches Anfeuern, obwohl alles unter enormem Zeitdruck geschieht. Man muss allerdings anmerken, dass die Experience des Bombenentschärfens damit weiter unterstützt wird.

8.1.2 It Takes Two

Anders ist es bei „It Takes Two“. Es überzeugt besonders durch sein abwechslungsreiches Gameplay und dem sehr guten Pacing. Es erfordert von beiden Seiten nicht sehr viel können, dadurch ist es für sehr viele Leute zugänglich und es lässt sich mit fast jeder Person spielen. Kommunikation ist sehr wichtig, da viele Rätsel lösungsorientiertes Denken beider Spielenden benötigen. Außerdem bietet das Spiel eine entspannende Atmosphäre, in der man sich schnell verliert.

8.2 Couchparty

8.2.1 Mario Party

"Mario Partyïst das klassische Party-Spiel. Wie für das Genre üblich bietet es eine enorme Vielfalt an Mini-Spielen und ein hohes Maß an Zufall, um auch die Schlechteren einer Gruppe im Spiel zu inkludieren.

8.2.2 Jackbox

Bei den Jackbox-Spielen handelt es sich um eine Reihe von verschiedenen Mini-Spielen, die einen oft an klassische Papier-Spiele erinnern, die man früher und auch heute noch ohne viel Aufwand gespielt hat (z. B. Wörter Raten). Es ist besonders gut darin lustige Geschichten oder andere Kreationen aus den Spielenden herauszuholen. Es bietet aber auch einige komplexere Spiele, die für eine erfahrenere Gruppe noch mehr Abwechslung bieten. Die Experience entsteht hier aber hauptsächlich durch die Imagination der Spielenden.

8.3 Couchparty VR-Games

Couch-Party VR-Games sind per Definition asymmetrisch. In dieser Kombination wird es schon schwieriger, gute Spiele zu finden, da der Markt noch klein ist und die großen Studios (Nintendo, EA, etc.) noch nicht in den Markt investieren, da es finanziell noch keinen Sinn macht.

8.3.1 Acron: Attack of the Squirrels!

Bei diesem Spiel spielt eine Gruppe aus 4 PC-Spielenden gegen einen VR-Spielenden. Ziel ist es, dem VR-Spielenden eine Eichel zu stehlen, um das zu verhindern, können VR-Spielende die Gruppe mit „Steinen“ bewerfen. Hier fehlen leider die Interaktionen zwischen den Spielenden in der Gruppe untereinander. Trotzdem kommt es bei der Community gut an, jedoch bietet es laut Reviews zu wenig Inhalt und Abwechslung für den verlangten Preis[34].

8.3.2 Takelings House Party

"Takelings"bringt in das Chaos von Couchparty-Spielen noch einen VR-Spieler. Es hat verschiedene Spielmodi, in denen das Ziel des VR-Spielenden es ist, alle „Takelings“ einzufangen. Leider gibt es auch hier wieder wenig Interaktion zwischen den PC-Spielenden.

Kapitel 9

Implementation

9.1 Was muss bei VR-Game-Design beachtet werden?

VR-Spiele bringen einige Limitationen mit sich, die es zu überwinden gilt. Bei der ersten Limitation handelt es sich um die Größe des Play-Spaces, also jenem physischem Raum, in dem sich VR-Spielende aufhalten. Das zweite Problem ist die Reisekrankheit (Motion-Sickness, eine körperliche Reaktion auf ungewöhnliche/unzusammenhängende Bewegung[20, S. 533]), vor allem bei Party-Spielen muss ihr Einfluss möglichst gering sein, da das Spiel von einer breiten Masse gespielt werden können soll. Die Reisekrankheit limitiert die Bewegungsmöglichkeiten der Spielenden und diese wiederum die Größe und Gestaltung der virtuellen Umgebung.

Außerdem muss die Größe des Spielenden (um sicherzustellen, dass alles, womit Spielende interagieren sollen, erreichbar ist) als auch Ängste wie zum Beispiel „Klaustrophobie“, beachtet werden. Im Falle von einem Party-Spiel spielt auch das technische Know-How eine Rolle, da man sehr schnell in das Spiel eintauchen können soll.

9.2 Stimmung des Spieles

Das im Zuge dieser Arbeit entwickelte Spiel soll eine bestimmte Stimmung erzeugen. Am Anfang des Design-Prozesses muss diese Stimmung festgelegt werden.

9.2.1 Core Pillars

Fast but Strategic

Das Ziel des Spiels soll es nicht sein, möglichst schnell Spells zu aktivieren. Es muss etwas Strategie dahinter sein, damit Spielende gefordert werden und eine wertvolle Entscheidung treffen können. Da es sich um ein Party-Spiel

und kein Strategie-Spiel handelt, sollte es möglich sein, schnelle Entscheidungen zu treffen. Um das zu ermöglichen, muss das Spell-System vergebend sein und muss sich auf die aktuelle Spielsituation anpassen können (mehr dazu in Kapitel 9.3.4).

Teamwork & Kommunikation

Außerdem muss Strategie und Absprache zwischen den PC-Spielenden angeregt werden, um weiter das Teamwork der Gruppe zu fördern. Wie am Anfang der Arbeit bereits definiert, braucht Teamwork ein gemeinsames Ziel. Das Ende der Strecke kommt hier recht schnell in den Sinn, allerdings ist das allein nicht genug, um Kommunikation zwischen den Spielenden zu erzeugen, da es zu weit in der Zukunft liegt und sich nicht verändert. Eine Lösung und mögliche Lösungsansätze für dieses Problem werden in Kapitel 9.3.4 besprochen.

God-Like

Vor allem der VR-Spieler soll sich mächtig fühlen. Erreicht werden soll dieses Gefühl durch den deutlichen Größenunterschied zwischen Magier und Snowboarder und Combos, also der Möglichkeit, mehrere Snowboarder gleichzeitig von der Strecke zu schmeißen. Es muss allerdings beachtet werden, dass das alleinige „auf die Strecke schmeißen“ keinen Spaß machen wird, wenn davor nicht etwas dafür getan werden musste. Hier kommt die „Lens of Challenge“[20] ins Spiel, man sollte besonders darauf achten, dass die Schwierigkeit, das Ziel zu erreichen, mit dem Skill des Spielenden übereinstimmt, außerdem ist es wichtig, die Interaktion nicht zu einseitig zu gestalten. Das Gefühl der Macht kann nur erzeugt werden, wenn Spielende denken, es war ihre alleinige Leistung.

9.2.2 Target Experiences

Da es sich bei dem Projekt um ein asymmetrisches Spiel handelt, wurden für die PC- und die VR-Spielenden jeweils andere Target Experiences festgelegt.

Snowboarder (PC)

Für die Snowboarder sind folgende zwei Experiences am wichtigsten: die (Snowboarding-) Action und das Teamwork der Gruppe. Mithilfe der „Lens of Essential Experience“[20, S. 55] kann versucht werden, diese Experiences zu erzeugen.

Um das Gefühl „Snowboarden“ akkurat wiederzugeben, wurden drei wichtige Komponenten gefunden: Tricks, Stürzen und Beharrlichkeit. Man könnte auch sagen, dass die Kälte ein Teil des Snowboardens darstellt, jedoch wurde dagegen entschieden, da es dem Core Pillar „God-Like“ widerspricht,

wenn sich die Snowboarder über die Körpertemperatur ihrer Charaktere Gedanken machen müssen.

Eine weitere wichtige Experience der Snowboarder ist das Teamwork sogar so wichtig, dass es, wie oben erwähnt, zu einem der Core-Pillars gemacht wurde. Wie Teamwork erzeugt werden soll, wird in Kapitel 9.3.4 behandelt.

Magier (VR)

Für den Magier sind die folgenden zwei Experiences am wichtigsten: Macht und Multitasking.

Diese göttliche/magische Kraft soll vor allem durch das Zaubern, also den Spells (siehe 9.3.2]) und die Aneinanderreihung und taktische („Fast but Strategic“) Platzierung dieser erzeugt werden. Sie wird auch über den visuellen Größenunterschied der Snowboarder und dem Magier dargestellt.

Multitasking ist vor allem im Bereich der Interaktionen wichtig, der Magier sollte immer schon bevor er einen Spell platziert hat, darüber nachdenken, was der nächste Zug sein wird. Um das zu erreichen, sollten Spells nicht immer sofort verfügbar sein. Ziel ist es, dadurch mehr Strategie in das Spiel zu bringen.

9.3 Zusammentreffen beider Welten (PC + VR)

9.3.1 Streckendesign

Der erste Punkt und auch Grundstein für das restliche Design ist das Konzept der Strecke. Es hat sowohl Einfluss auf die PC-Spielenden als auch auf den VR-Spielenden, da es den Ort darstellt, in dem beide miteinander Interagieren.

Limitationen

Der VR-Spielende muss die Möglichkeit haben, zu jedem gegebenen Zeitpunkt mit einem Großteil der Strecke zu interagieren (Fallen platzieren etc.). Außerdem ist es wichtig, dass sich Spells immer in Reichweite befinden. Die Strecke muss jedoch auch (aufgrund anderer technischer Limitationen) zu jedem Zeitpunkt komplett geladen und sichtbar sein.

Das Streckendesign muss es zulassen, die Strecke so anzupassen, dass eine Fahrt von oben nach unten ca. drei Minuten dauert. Des Weiteren muss die Breite der Strecke genug Platz, um Hindernissen auszuweichen und den Spielenden eine gewisse Toleranz für Fehler beim Lenken der Snowboarder lassen.

Zur Auswahl stehen drei verschiedene Streckendesigns, deren vor und Nachteile behandelt werden und begründet wird, warum schlussendlich für Streckendesign Nr. 3 entschieden wurde. Alle drei Designs versuchen die Be-

wegung des VR-Spielenden zu vermeiden, um das Gefühl der Reisekrankheit zu mindern.

Designoption 1 - Spirale

Die Snowboarder fahren in diesem Design in einer Spirale um den VR-Spielenden herum. Es ermöglicht eine freie Strecke für die Snowboarder, jedoch ist wenig Platz für VR-Spielende. Dieses Design ist auch sehr unrealistisch, da es sich hier nicht um eine klassische Ski-Strecke handeln kann, da Teile „Schweben". Dieser Effekt wird verstärkt dadurch, dass es sich für die Spielenden so anfühlt, als wären sie in einer Wasserrutsche. Das widerspricht der Target-Experience des Snowboardens.

Designoption 2 - Kristallkugel

Dieses Design bringt auf der Seite der Snowboarder und von dem VR-Spielenden deutliche Verbesserungen. Da VR-Spielende durch eine Kristallkugel auf die Snowboarder herabschaut, kann die Strecke unglaublich frei sein, es kann harte Kurven und Richtungswechsel geben. Der VR-Spielende hat die Möglichkeit, einen eigenen Raum zu bekommen, indem der Zugang zu den Spells einfach ist. Der größte Nachteil ist hier die Technik, die im Hintergrund gebraucht wird. Technisch ist es sehr schwer umzusetzen, da die Welten des VR-Spielenden und der Snowboarder voneinander getrennt und über die Kristallkugel wieder zusammengebracht werden müssen.

Designoption 3 - Berg

Die Idee des Berges wurde am Anfang etwas vernachlässigt, da sie sehr eingeschränkt erschien. Jedoch bringt der Berg neben der technisch um einiges leichteren Umsetzung wieder etwas Realismus und eine schöne Atmosphäre ins Spiel, die die Experience des Snowboardens verstärken kann. Aus der Sicht des VR-Spielenden, kleinen Berg, in der Mitte des VR-Play-Spaces, können die Snowboarder spiralförmig hinunterfahren. Wie bei der Spirale hat man hier aber auch das Problem, dass die Snowboarder permanent in die Kurve lenken müssen, dieses Problem wird in Kapitel 9.3.3 weiter behandelt. Der Berg kann automatisch mit den Snowboardern mitgedreht werden, damit sie immer im Sichtfeld des Magiers sind. Der Magier soll aber auch eine Möglichkeit haben, diese Drehung zu überschreiben, damit er auch Teile der Strecke die außerhalb des interagierbaren Bereiches liegen beeinflussen kann.

9.3.2 Interaktionsdesign

Da der Wettbewerb zwischen PC und VR asymmetrisch ist, müssen beide Parteien die jeweils andere Seite beeinflussen können, um es interessant zu

machen. Der Magier muss durch Spells und andere Tools die Snowboarder behindern können und die Snowboarder müssen Einfluss auf die Welt des VR-Spieleden haben.

Spell Design

Um dem VR-Spielenden das Gefühl von Zaubern zu übermitteln, braucht es neben der Möglichkeit zu Zaubern auch gute Interaktionen. Das Spell-System muss dem Magier die Möglichkeit geben, zu Multitasking und Kombis zu machen, damit das gefühlt der Macht aufkommt. Da der VR-Spielende zwei Hände hat, ist es naheliegend, dass beide Hände an unterschiedlichen Spells arbeiten können. Um eine gewisse Vielfalt in die möglichen Spells zu bringen, wurden drei Spell-Kategorien eingeführt.

Instant-Spell Sollen einen sofortigen Effekt auf die PC-Spielenden haben. Instant-Spells sollen sich besonders gut für die Kombination mit anderen Spell-Typen eignen. Sie dürfen auch in die Masse der Snowboarder platziert werden, da sie niemals sofort töten dürfen. Ein Beispiel hierfür wäre eine Eisfläche, die die Snowboarder zum Rutschen bringt.

Short-Term-Spell Dieser Spell-Typ braucht eine gewisse Zeit, um sich zu aktivieren. Somit müssen Spielende sich vorher etwas genauer überlegen, wo sie am besten zu platzieren sind, und die Snowboarder haben etwas Zeit, um auszuweichen. Short-Term-Spells dürfen auch tödlich enden, ein Beispiel hierfür wäre eine Bombe, die die Snowboarder in die Luft schleudert. Alle Short-Term-Spells müssen nach einer gewissen Zeit wieder verschwinden.

Long-Term-Spell Das Ziel von Long-Term-Spells ist es, den Spielenden die Möglichkeit zu geben, etwas strategischer zu spielen. Sie können nur begrenzt und an vordefinierten Orten aktiviert werden. Ziel ist es, Spielende dazu zu bewegen, zu Beginn einer Runde abzuwägen, einen Long-Term-Spell einzusetzen und dafür weniger Instant- und Short-Term-Spells, oder darauf zu verzichten, um flexibler zu sein. Beispiele hierfür wären das Absperren von gewissen Streckenteilen oder das zum Einsturz bringen von Bäumen oder Geröll auf der Strecke.

Balancing-Cooldown Die Anzahl der Spells, welche der Magier verwenden kann, muss limitiert sein. Der einfachste Weg, um Spells zu limitieren, ist ein Cooldown, welcher verhindert, dass zwei Spells direkt nacheinander platziert werden können. Es wird jedoch bei diesem System schnell dazu kommen, dass VR-Spielende immer den „Stärksten“ Spell nehmen und es dadurch schwer zu Balancen ist.

Balancing-Mana Bevor ein Spell platziert werden kann, muss er in einen Mana-Kessel getaucht werden, um ihn zu aktivieren. Dieser Mana-Kessel kann nur eine begrenzte Menge an Mana halten und befüllt sich über den Verlauf des Spieles von selbst. Ein Mana-Kessel bietet gleich mehrere Vorteile: Balancing und Physische-Interaktion. Mana kann dabei helfen, Speils untereinander zu Balancen, indem ihre Kosten erhöht oder verringert werden. Außerdem fügt der Kessel einen Zwischenschritt hinzu, das macht die Interaktion komplexer und interessanter. Sobald ein Spell einmal aufgeladen ist, ist man dazu gezwungen, ihn auch zu platzieren, da er sonst seine Ladung wieder verliert.

Die Anzahl der Spielenden, die noch im Rennen sind, könnte sich auf die Mana-Regeneration auswirken, so entsteht ein negativer Feedbackloop (siehe 9.3.4), es wird für den Magier schwieriger zu gewinnen, wenn er kurz davor ist, den letzten Snowboarder zu Fall zu bringen. Das erzeugt einen weiteren Pacing Spike wie bei einem Bosskampf am Ende einer Runde und hat einen automatischen Balancing[31, S. 296]-Effekt.

Balancing-Interaction Ein weiterer Weg, wie Spells gebalanced werden könnten, wäre durch die Dauer / Komplexität der Interaktion. Ein großer Vorteil hierbei wäre, dass es den Spielenden etwas natürlicher vorkommt als ein künstlicher Cooldown, dadurch sollte es auch weniger frustrierend sein, wenn gerade kein Spell bereitsteht, da der Magier selbst dafür verantwortlich war, sie herzustellen. Den Spielenden soll dadurch nahegelegt werden, dass sie es beim nächsten Mal durch verbessern, ihrer Skills, besser machen können, das nennt man auch interne Attribution[35].

9.3.3 Playercontroller & Obstacles

Tricks

Tricks können zum einen dazu genutzt werden, ein Risiko einzugehen, um die Respawn-Zeit zu verkürzen oder als eine Movement-Mechanik, die dabei hilft, Obstacles auszuweichen. In das Spiel wird vermutlich eine Mischung beider Mechanics implementiert.

Respawns

Das Wiedereintreten von ausgeschiedenen Spielenden ist eine wichtige Mechanik, um das Gefühl des Snowboardens einzufangen und um den Party-Aspekt des Spieles zu fördern. Es gibt verschiedene Möglichkeiten zu regeln, wann Spielende wieder in die Runde eintreten.

Timer Eine sehr simple Möglichkeit wäre, Ausgeschiedene nach einer gewissen Zeit automatisch wieder dem Rennen hinzuzufügen. Das hätte den Vorteil, dass es zum einen sehr leicht umsetzbar ist und zum anderen den

Spielenden eine kleine Pause verschafft und der Timer sich durch andere Faktoren beeinflussen ließe, was wiederum beim Balancing helfen kann.

Sekundäres Ziel für ausgeschiedene Spielende Ein weiterer Weg, der die ausgeschiedenen Spielenden besser ins Spiel inkludiert, wäre ein sekundäres Ziel. Sie müssten, um wieder ins Rennen zu kommen, eine kleine Aufgabe lösen, bestimmte Knöpfe schnell hintereinander drücken (verbunden mit viel Game-Juice, um die durch die Spielenden wahrgenommene Intensität des Gameplays zu steigern), oder sie könnten den VR-Spielenden behindern, mehr dazu in Kapitel 9.3.4. Dieser Möglichkeiten könnte wieder dabei helfen, den Spielenden ein Gefühl von Kontrolle zu geben. Ähnlich wie bei dem Magier (9.3.2) ist es auch auf der Seite der Snowboarder hilfreich, wenn ein ausgeschiedener Spielender die Schuld für die schlechte Team-Performance in sich selbst sucht (interne Attribution) und sich deshalb versucht zu verbessern.

Automatische Lenkhilfe

Da die Spielenden bei dem Berg- oder Spiralen-Streckendesign permanent im Kreis lenken müssen, wäre es wichtig, dem entgegenzuwirken, um das Gefühl des „im Kreis“-fahrens zumindest etwas zu mindern. Hier wäre eine automatische Lenkhilfe, die den Snowboarder automatisch nach unten fahren lässt, die einfachste Methode. Verbunden mit einer Lenkung, welche relativ zur Kamera funktioniert, haben Spielende eine angemessene Lenk-Freiheit in beide Richtungen.

Obstacles

Obstacles können sowohl von der Strecke vorgegeben sein als auch durch den Magier platziert werden. Sie zwingen die Snowboarder, Entscheidungen zu treffen, insbesondere im Falle von *Positive Obstacles*. Normale Obstacles sollten von den Snowboardern vermieden werden.

Positive Obstacles *Positive Obstacles* sind ähnlich wie Pick-Ups in z. B. *Mario Cart*, nur sind sie in die Welt integriert. Sie haben einen positiven Effekt auf die Gruppe, jedoch werden sie von VR-Spielenden eher angegriffen/beschützt.

Spieler führen

Wenn man Spielende für riskantes Fahren belohnt, entsteht eine Meaningful-Decision. Es entsteht eine Risk-Reward Situation, in der sich schlechte Teammitglieder von den Guten abspalten und andere Wege nehmen. Mithilfe von Rampen und Slalom-Bögen, die die Respawn-Zeit verkürzen, kann man riskantes Fahren fördern.

9.3.4 Teamwork erzeugen - Krone & andere Lösungsansätze

Um die Snowboard-Gruppe dazu zu bewegen, zu kooperieren, müssen sie ein gemeinsames Ziel haben. Dieses Ziel ist schlussendlich das Ende der Strecke jedoch braucht es auch ein kurzzeitiges Ziel wie zum Beispiel einem ausgeschiedenen Spielenden dabei zu helfen, wieder ins Rennen zu kommen. Mögliche Aufgaben dafür wären das Einsammeln von bestimmten Punkten auf der Strecke, das Halten der Krone oder eine gemeinsame Aktion (z. B. einen Trick machen).

Coop-Points

Bei Coop-Points müssen verschiede Spielende zur selben Zeit an zwei verschiedenen Orten sein. Dadurch muss es zwingend zu einer Absprache zwischen den Spielenden kommen, um auszumachen, wer wohin fährt.

Krone

Der erste Spielende hält immer die Krone. Wenn die Krone länger als eine gewisse Zeit vom selben Spielenden gehalten wird, kann ein ausgeschiedener Spielender wieder auf die Piste. Sollte die Krone von einem anderen Spielenden übernommen worden sein, bevor ein Snowboarder respawnt ist, wird der Timer wieder länger. Nach einem Respawn, muss die Krone an einen anderen Spielenden übergeben werden, um Teamwork und Kommunikation innerhalb des Teams zu fördern.

Gemeinsame Aktion

Eine gemeinsame Aktion wäre beispielsweise, dass alle Snowboarder zur selben Zeit Springen oder einen Trick ausführen. Das würde wieder ein großes Risiko mit sich bringen, da alle gleichzeitig sterben könnten.

Negativer Feedbackloop

Ein negativer Feedbackloop versucht Änderungen entgegenzusteuern um so ein System stabil zu halten[31, S.133]. Snowboarder, die weiter hinten sind, sollten schneller werden, damit sie mit der Gruppe mithalten können. Wie in 9.3.2 angesprochen, ist es auch wichtig, PC und VR untereinander zu Balancen. Neben dem bereits angesprochenen gibt es noch folgende weitere Möglichkeiten: Ausgeschiedene Spielende könnten den VR-Spieler daran hindern, weitere Spells zu aktivieren oder gute Sicht auf die Strecke durch z. B. Nebel verhindern. Spells könnten auch dynamisch gestärkt oder geschwächt werden, indem man deren Range oder Dauer verändert.

Kapitel 10

Post-Mortem

Im Designprozess ist nicht immer alles so, wie man es erwartet, oft kommt es zu unerwarteten Nebeneffekten, die man im ursprünglichen Design nicht beachten konnte. Deswegen ist die Flexibilität aller Komponenten, um späteres Balancing zu ermöglichen und besonders Playtesting in verschiedenen Phasen des Designprozesses notwendig.

Abbildung 10.1: Screenshot einer Situation im Spiel aus der Sicht der PC-Spielenden

10.1 Playtests

Durch Playtests wurden einige Mängel im Design gefunden, die dann auch rechtzeitig behoben werden konnten. Leider sind Playtests erst spät in der Entwicklung sinnvoll gewesen. Außerdem wurden neue Bereiche aufgedeckt, die vorher vernachlässigt wurden. Sollte das Spiel weiterentwickelt werden, sind Playtests ein wichtiger Punkt, um die Qualität des Spieles zu sichern und weiter zu polishen, da die Grundsysteme mittlerweile funktionieren.

Feedback

Einige Dinge, auf die wir durch Playtests aufmerksam wurden, wurden bereits implementiert. Darunter ist auch mehr Feedback für tote Spielende und wann sie wieder Respawnen können. Neben dem initialen Feedback für Tod und Respawn, ist es auch wichtig, den Spielenden über eine gewisse Zeit hinweg zu sagen, dass sie eine bestimmte Aktion ausführen können, sollten sie den ersten „Tipp“ übersehen haben. Hier haben wir uns für eine leichte Vibration des Controllers entschieden (Diese Idee kam nur dank des Feedbacks unserer Playtester). Es ist manchmal aber auch wichtig, den Spielenden Zeit zu geben undsie selbstständig entscheiden zu llassen, wanneine Aktion ausgeführt wird, anstatt ihnen einfach zu sagen „Jetzt!“. Neben Feedback ist also auch Agency (Die Möglichkeit der Spielenden Einfluss auf das Geschehen in der Spielwelt zu haben[31, S.98]) wichtig.

Eine weitere Sache, auf die wir bei Playtests aufmerksam wurden, war, wie die Spielenden mit Spells interagieren. Leider wurde darüber anfangs weniger nachgedacht. Es wurde jedoch schnell klar, dass viele Spielende nicht intrinsisch herausfinden konnten, wie Spells zu aktivieren sind. Es wurden verschiedene Möglichkeiten ausprobiert, um es klarer zu gestalten. Das ursprüngliche Aktivieren per Knopfdruck wurde durch ein Collider basiertes Aktivieren ausgetauscht. Schlussendlich wurde sogar der Collider der gehaltenen Spells verkleinert, um den Spielenden die nötige Genauigkeit bei der Platzierung der Spells zu geben.

Balancing

Außerdem halfen Playtests enorm beim Balancing des Spieles. Die ersten Settings, die bei beispielsweise der Geschwindigkeit der Snowboarder oder der Menge an Mana eingestellt wurden, waren immer falsch und mussten immer leicht angepasst werden, um das Spiel möglichst fair zu halten. Auch das ist nur dank Playtests möglich.

Positives

Neben den Dingen, die nicht gut funktioniert haben, hat das Konzept der Krone, wie erwartet, sehr gut für Teamwork innerhalb der Gruppe gesorgt. Auch die Tricks und *Positive Obstacles* (siehe 9.3.3) haben den gewünschten Effekt, riskantes Fahren zu fördern. Die verschiedenen Spell-Typen wurden von den VR-Spielenden wie erwartet kombiniert, da sie gemeinsam einen größeren Effekt auf die Snowboarder hatten.

10.2 Kommunikation

Kommunikation innerhalb des Teams ist größtenteils problemlos verlaufen. Aufgrund der Größe des Teams und der Aufgabenverteilung konnte effizi-

ent kommuniziert werden. Besonders in der Ideenfindung war eine offene Kommunikation vorhanden, in der jeder etwas einbringen durfte. Zu einigen Feature-Drafts wurden interne Dokumente erstellt, damit sie besser kommuniziert werden konnten. Allerdings ist ein Großteil der Kommunikation verbal abgelaufen. Das hat in manchen Fällen dazu geführt, dass einige Dinge mehrmals erklärt werden mussten. Obwohl das durch die genauere Führung von Dokumenten hätte verhindert werden können, wäre es vermutlich den zeitlichen Mehraufwand dank der geringen Größe unseres Teams nicht wert gewesen. Allerdings ist es bei längeren Projekten oder größeren Teams durchaus empfehlenswert möglichst viel zu dokumentieren, damit Informationen nicht wiederholt werden müssen, und Entscheidungen rückverfolgt werden können.

10.3 Zukunft des Projektes

Sollte das Projekt weiterentwickelt werden, ist vor allem mehr auf das Design der Interaktionen des VR-Spielenden zu achten. Derzeit sind alle Interaktionen sehr statisch und es sollten, um das Rebalancing der Spells für die Spielenden verständlicher und auch abhängiger von den Skills des Magiers zu machen, komplexere Interaktionen (siehe 9.3.2) implementiert werden. Leider waren die dafür benötigten Systeme zu komplex, um sie in der kurzen verbleibenden Zeit noch umzusetzen, und es wurde mehr Zeit in das Polishing von dem, was bereits da war investiert.

Grundsätzlich war der Grundgedanke, so wenig wie möglich zu implementieren, dafür mehr darauf zu achten, das Implementierte möglichst zu perfektionieren, eine der besten Entscheidungen im Designprozess.

Farb- und Formpsychologie in asymmetrischen VR-Spielen

Emma Wehinger

D I P L O M A R B E I T

eingereicht am
Ausbildungsschwerpunkt
Mediendesign - Gamedesign

betreut durch
Alexander Hager

im März 2023

Kapitel 11

Überblick über die Materie

11.1 Farbpsychologie

11.1.1 Was ist Farbe?

Wissenschaftlich betrachtet, sind Farben Eindrücke, die unsere Sinne durch die Augen und das Gehirn vermittelt bekommen. Farben entstehen durch die Absorption und Reflexion, die das Licht auf Oberflächen wirft. Das menschliche Auge ist für die Sinnesempfindung von Farben verantwortlich, die dann wiederum im Gehirn verarbeitet werden. Man kann Farbe auch als Eigenschaft des Lichts betiteln. Prinzipiell nimmt jeder Mensch Farben und Farbtöne anders wahr. Welche Farbe wahrgenommen wird, hängt von dem Gegenstand ab, der Lichtquelle und den Augen des Betrachtenden. Licht besteht aus unterschiedlichen Wellenlängen, je nach Länge der Welle nimmt das Auge eine andere Farbe war. Schwarz ist keine Farbe, denn sie kann auch in Abwesenheit des Lichts und völliger Dunkelheit existieren. [36] [37]

Weißes Licht kann mithilfe eines Prismas in Spektralfarben gebrochen werden. Spektralfarben sind diejenigen Farben, die für das menschliche Auge einen sichtbaren Farbeindruck und somit einen Farbton hinterlassen. Insgesamt gibt es sechs verschiedene Spektralfarben. Spektralfarben werden umgangssprachlich auch „Regenbogenfarben“ genannt. Zu ihnen zählen Rot, Orange, Gelb, Grün, Blau, Indigo und Violett. Die durch die Brechung von Licht entstandenen Farben können nicht in weitere Farbtöne zerlegt werden. Das Licht kann manchmal auch infrarotes Licht oder ultraviolettes Licht enthalten, dass für das bloße Auge nicht sichtbar ist. Wenn man jedoch gelbes Licht oder jede andere Spektralfarbe durch ein Prisma fallen lässt, bekommt man dieselbe Farbe raus, die man im Prisma brechen wollte. Man kann als Schlussfolgerung daraus ziehen, dass weißes Licht das Einzige ist, was durch ein Prisma in Spektralfarben gebrochen werden kann. [36] [37]

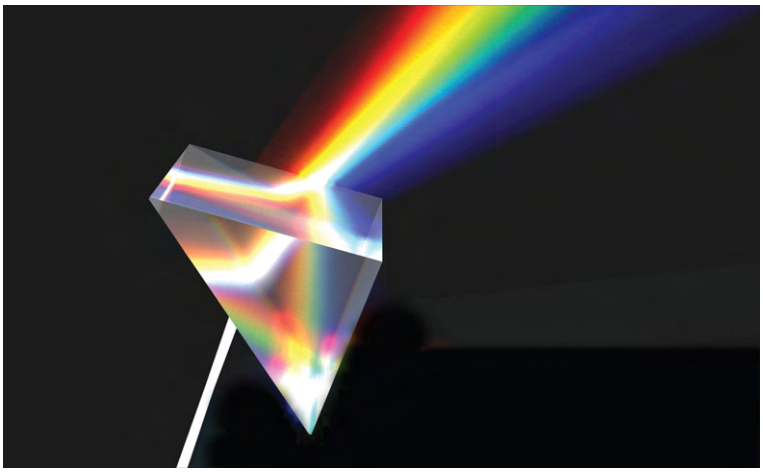


Abbildung 11.1: Farbprisma[36]

In der Kunst ist Farbe das beste Mittel, um visuell Emotionen auszudrücken oder Emotionen bei Menschen zu erzeugen. Farben sind ein gutes Instrument, wenn es darum geht, bei einem Menschen Gefühle und Emotionen auszulösen. Aber nicht nur dass, es wird dem Betrachtenden dadurch auch eine bestimmte Stimmung vermittelt. Künstler können dadurch gezielt bei Menschen bestimmte Gefühle auslösen und dies alleine durch das Einsetzen von bestimmten Farben und Farbtönen. [36] [38] [39]

Dabei muss beachtet werden, dass Farben in verschiedenen Kulturen verschiedene Bedeutungen haben und die Symbolik hinter manchen Farbtönen in bestimmten Ländern und Kulturkreisen eine andere ist. Doch grundsätzlich kann man jeder Farbe eine bestimmte Bedeutung und Emotion zuschreiben, die vielleicht in manchen Kulturen ein wenig abweicht, aber im Grunde genommen immer sehr ähnlich ist. [36] [38] [39]

Man kann klar die kulturelle Bedeutung der Farben von der psychologischen Bedeutung trennen. Die wichtigsten Farben für Kunst und Design sind die Farben Blau, Gelb, Grün, Orange, Rosa, Rot, Violett, Grau und Braun. Jede dieser Farben hat eine andere Wirkung auf die Gefühle und Emotionen der Betrachtenden. Der psychologische Effekt hinter der Wirkung einer Farbe hat rein gar nichts mit der kulturellen Interpretation dieser zu tun. [36] [38] [39]

Es gibt eine endliche Anzahl an Variationen, die eine Farbe für das menschliche Auge annehmen kann. Grundsätzlich gibt es eine unendliche Anzahl an Schattierungen, die eine Farbe haben kann, wobei die meisten für das Sinnesorgan eines Menschen nicht sichtbar sind. Die Schattierung einer Farbe ändert prinzipiell den psychologischen Effekt nicht. [36]

11.1.2 Der Farbkreis und Farbschemas

Der Farbkreis besteht normalerweise aus 12 Farben, er kann aber auch aus 24 und manchmal aus bis zu 48 Farben bestehen. Das hängt aber von dem Ersteller des Farbkreises ab. Das grundlegende Farbrad besteht aber jedoch nur aus 12 Farben. Um den Farbkreis besser zu verstehen, muss man sich zuerst die additive und die subtraktive Farbmischung ansehen. [36] [38] [39]

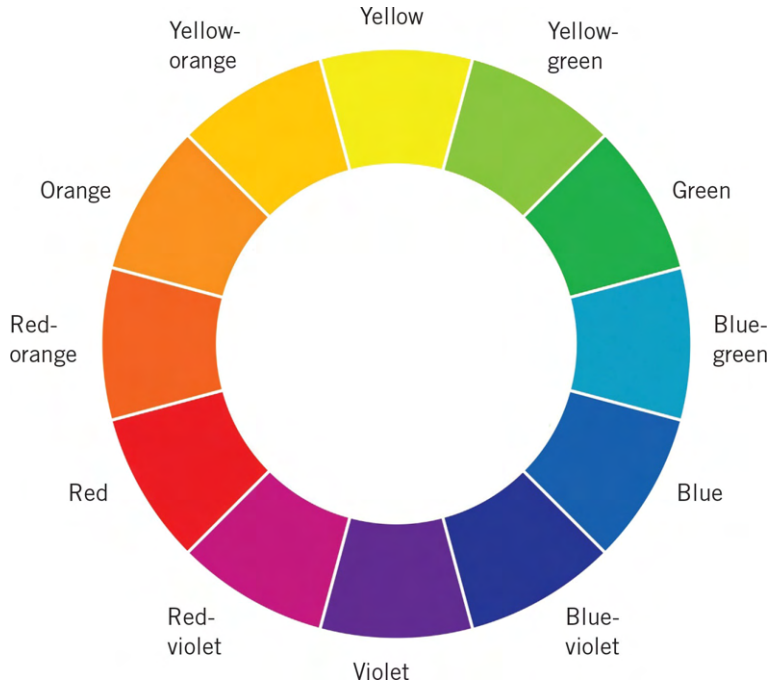


Abbildung 11.2: Farbrad[36]

Farben können in Primär -, Sekundär -und Tertiärfarben aufgeteilt werden. Dabei unterscheidet man zwischen der additiven Farbmischung und der subtraktiven Farbmischung. [36] [38]

Bei der subtraktiven Farbmischung, auch CMYK-Modell genannt, sind die Primärfarben Gelb, Magenta und Cyan, sie können nicht aus anderen Farben gemischt werden. Jedoch kann aus ihnen jede existierende Farbe erstellt werden. Die Sekundärfarben in diesem Modell sind Rot, Grün und Blau und die Tertiärfarbe ist Schwarz. Bei dieser Farbmischung entstehen Farbeindrücke dadurch, dass aus bereits existierendem Licht das Licht einzelner Spektralfarben herausgefiltert wird. [36] [40] [41]

Die additive Farbmischung, auch RGB-Modell genannt, besteht aus den Lichtfarben Rot, Grün und Blau. Rot, Grün und Blau sind in diesem Modell die Primärfarben, also Grundfarben. Die Sekundärfarben sind Gelb, Magenta und Cyan. Die Tertiärfarbe ist Weiß. Diese entsteht durch die Mischung

der Grundfarben Rot, Grün und Blau. Bei dieser Farbmischung wird zu vorhandenem Licht das Licht weiterer Spektralfarben hinzugefügt. Mischt man das Licht dieser Grundfarben passend zusammen, kann man fast jeden Farbeindruck erschaffen. [36] [40] [42]

Im grundlegenden Farbkreis gibt es einen Unterschied bei den Primär -, Sekundär -und Tertiärfarben zum CMYK - und RGB Modell. Im Farbrad mit 12 Farben sind die Primärfarben Rot, Gelb und Blau. Die Sekundärfarben sind die Farben, die durch das Mischen von zwei Primärfarben entstehen, das sind Farben wie Orange, Grün und Violett. Die Tertiärfarben sind die Farben, die beim Mischen von einer Primärfarbe und einer benachbarten Sekundärfarbe entstehen. Die Tertiärfarben sind all jene Farben, die die noch vorhandenen Lücken im Farbkreis füllen. [36] [40] [38]

Die wichtigste Aufgabe des Farbrads ist es viele Farben auf einmal zu sehen, um sie dadurch besser zu verstehen und zu lernen, Farben zu wählen die gut zueinander passen. Es zeigt uns visuell das Verhalten der Farbtöne zueinander, dadurch kann man harmonische und ästhetische Kombinationen bilden. Außerdem gibt es fünf verschiedene Farbschemen, die einem helfen harmonische und dynamische Atmosphäre in seinen Kompositionen zu kreieren. [36] [39]

Beim komplementären Farbschema wählt man die Farben aus, die einander gegenüber liegen. Komplementäre Farben wirken nebeneinander viel intensiver und heller als Farben, die im Farbrad benachbart sind. Ein Beispiel für Komplementärfarben wären zum Beispiel Rot und Grün, sie liegen im Kreis gegenüber. [36]

Das triadische Farbschema besteht aus einem gleichseitigen Dreieck. Dieses Modell fokussiert sich auf eine dominante Farbe und zwei zueinander sehr harmonische Farben. Ein Beispiel dafür wäre die Kombination aus den Farbtönen Violett, Grün und Orange. Diese Farben erzeugen sehr viel Kontrast, wirken aber sehr harmonisch auf das menschliche Auge. [36]

Beim tetraedrischen Farbschema besteht die Kombination der Farben aus Farbtönen, die an den Ecken des Rechtecks oder Quadrats liegen. Dieses Schema besteht aus jeweils 2 Paaren von Komplementärfarben, die zwar zueinander viel Kontrast haben, aber im Gesamtbild keine dominante Farbe besitzen. Die Kombination soll ein harmonisches Bild erzeugen. [36]

Beim analogischen Farbschema werden meistens 2 Farben ausgewählt, die einer anderen Farbe benachbart sind. Diese Kombination von Farbtönen soll eine Einheit erzeugen, um das Gesamtbild ausgewogen wirken zu lassen. Ein Beispiel für solch ein Schema wären die Farben Rot, Rot-orange und Orange. [36]

Das letzte Farbschema ist das komplemenänter-geteilte Schema, es besteht aus eine Primär -oder Sekundärfarbe und zwei Tertiärfarben, die der Komplementärfarbe der Hauptfarbe benachbart sind. So eine Farbkombination wäre zum Beispiel Rot, Gelb-Grün und Blau-Grün. [36]

11.1.3 Farbtemperatur

Die Farbtemperatur ist grundsätzlich der Farbton, den eine Farbe annimmt, wenn sie von einer Lichtquelle beleuchtet wird. In der Kunst bezieht sich die Farbtemperatur auf das Gefühl, das der Betrachtende von bestimmten Farbtönen oder Gruppen von Farben bekommt. Dabei unterscheidet man zwischen zwei Gruppierungen, den warmen Farben und den kalten Farben. Warme Farben wären zum Beispiel Rot, Gelb und Orange. Kalte Farben wären unter anderem Blau, Violett und Grün. Während warme Farben auf den Betrachtenden so wirken, als würden sie auf einen zugehen, wirken kalte Farben so, als würden sie sich vom Betrachtenden entfernen. Wenn man Tiefe erzeugen will oder ein Bild dynamisch wirken lassen möchte, sollte man dafür warme und kalte Farben verwenden. Man kann diese auch klar im Farbkreis trennen, indem man zwischen Gelb und Violett einen Strich durch die Mitte zieht, als Ergebnis hat man auf einer Seite die warmen Farben und auf der anderen Seite die Kalten. [36] [38]

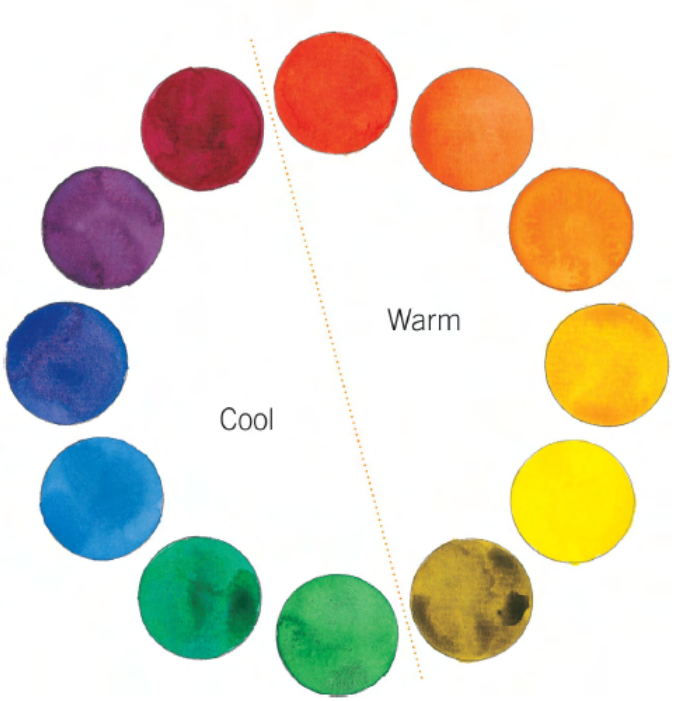


Abbildung 11.3: Farbrad in warme und kalte Farben unterteilt[36]

11.1.4 Farbrelativität

Prinzipiell werden Farben in warme und kalte Farben eingestuft, sie können aber auch in relativ warme oder kalte Farben innerhalb ihres Farbtons eingestuft werden. Gelb gehört zu den warmen Farben, mischt man Blau hinzu, wird der Farbton eher kühler. Mischt man hingegen Rot hinzu, wird der Farbton wärmer. Eigenschaften einer Farbe sind immer relativ zu ihrer Umgebung. Man kann Farben heller oder dunkler und wärmer oder kälter wirken lassen, indem man sie neben bestimmten Farben platziert. [36] [38] [39]

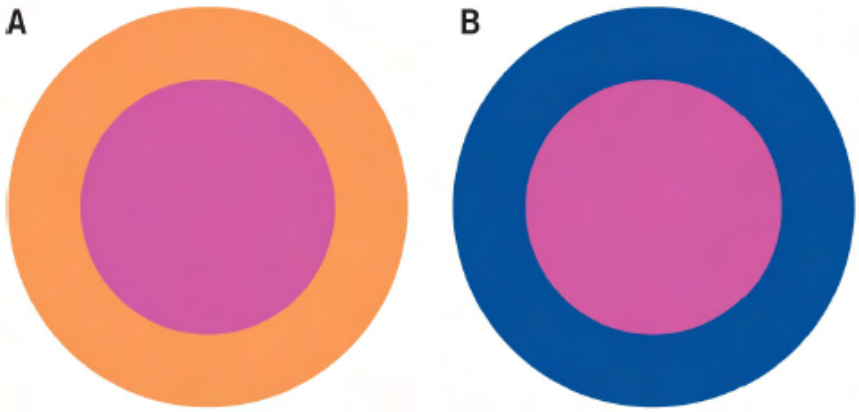


Abbildung 11.4: Farbrelativität zwischen zwei Farben[36]

11.1.5 Farbsättigung

Die Farbsättigung einer Farbe kann an ihrer Reinheit und Leuchtkraft gemessen werden. Wenn eine Farbe weder in ihrer Reinheit noch in ihrer Leuchtkraft gesteigert werden kann, ist sie gesättigt. Die Farbsättigung beschreibt also das Verhältnis, das die Farbigkeit zur Helligkeit eines Farbtons hat. Die Farbsättigung wird auch „Intensität“ oder „Chroma“ einer Farbe genannt. Schwarz, Weiß und Grau haben eine Sättigung vom Wert null, sie haben somit keine Farbsättigung. [36] [38]

11.1.6 Farbe, Licht und Schatten

Mit Farbe, Licht und Schatten kann die Stimmung und die Dramatik eines Bildes beeinflusst werden. Um zu verstehen, wie Licht und Schatten Objekte beeinflussen, muss das Objekt in drei Teile geteilt werden. Das Objekt wird in Licht, lokale Farbe und Schatten eingeteilt. Die lokale Farbe bezieht sich auf die tatsächliche Farbe eines Objekts ohne Einfluss von Licht und Schatten. Es gibt verschiedene Lichtquellen, mit denen Objekte beleuchtet werden

können. Darunter fallen die Sonne, der Mond und künstliches Licht. Wenn die Sonne ein Objekt beleuchtet, werden die lokalen Farben des Objekts wärmer und die Schatten kühler. Wenn der Mond ein Objekt beleuchtet, werden die lokalen Farben des Objekts kühler und die Schatten wärmer. [36] [38] [39]

11.1.7 Wie wirkt sich Farbe auf unsere Stimmung aus?

Farbpsychologie beschreibt, wie Farben die Stimmung und das Verhalten von Menschen beeinflussen. Wie Menschen Farben wahrnehmen, hängt von ihrem Geschlecht, ihrem Alter, ihrer Kultur und ihrem ethnischen Hintergrund ab. Dennoch gibt es viele universelle Reaktionen auf bestimmte Farben und Farbgruppen. [36] [38]

Gelb steht für Wärme, Glück, Hoffnung und Positivität. Die Farbe Gelb wird oft auch mit der Sonne oder dem Sonnenschein in Zusammenhang gesetzt.[36][43] Rot wird oft mit Feuer und Blut assoziiert und sie steht für Energie, Macht, Liebe und Passion. Es wurde erwiesen, dass die Farbe Rot den Blutdruck und den Puls erhöht. [36][43] Die Farbe Pink oder Rosarot wird mit Liebe und Romantik in Verbindung gesetzt. Die Farbe hat einen beruhigenden Effekt, wobei zu viel für den Menschen anstrengend sein kann.[36] [43] Blau wird mit dem Himmel und dem Wasser assoziiert. Die Farbe löst beim Betrachtenden Gelassenheit aus und wirkt entspannend. Dunklere Blautöne werden jedoch mit Verzweiflung und Traurigkeit in Zusammenhang gesetzt. [36] [43] Violett wird mit dem Adel verbunden, denn früher konnten sich nur sehr wohlhabende Menschen das violette Farbpigment leisten. Die Farbe Violett steht für Eleganz, Würde und Raffinesse. [36] [43] Schwarz wird mit dem Tod, der Angst, dem Bösen und Negativität assoziiert. Die Farbe Schwarz wird oft verwendet, damit andere Farben und Farbtöne mehr herausstechen. [36] [43] Weiß symbolisiert Reinheit, Unschuld und Göttlichkeit, sie wird auch als Farbe der Perfektion angesehen. [36] [43] Die Farbe Grün ist die Farbe der Natur. Sie wird mit Harmonie und Fruchtbarkeit assoziiert. Grün ist für das menschliche Auge die erholsamste Farbe. [36] [43] Orange steht für Enthusiasmus, Kreativität und Belebung. Das menschliche Auge nimmt Orange als die wärmste Farbe wahr. [36] [43]

11.2 Formpsychologie

In der Formpsychologie geht es darum, wie unser Gehirn versucht, einfache und stabile Formen herauszufiltern, die ihm vertraut sind. Das Gehirn versucht Elemente, die zusammengehören zu verbinden. Formen, die sich sehr ähnlich sind, werden in eine Gruppe gegeben. Wenn unser Gehirn die Entfernung nicht verwenden kann, versucht es, ähnliche Formen zu gruppieren. Es wird immer versucht, Formen zu vervollständigen. [39] [38]

11.2.1 Gestalt und Form

Die Kommunikation mit Formen taucht in allen künstlerischen Disziplinen auf und ist erstaunlich universell. Die drei Grundformen sind der Kreis, das Quadrat und das Dreieck. Der Kreis steht für Freundlichkeit und Positivität. Das Quadrat soll Vertrauen, Stabilität und Sicherheit kommunizieren. Das Dreieck steht für Gefahr und Aggressivität. Man kann die Formen von links nach rechts anordnen, in eine „Skala der Emotionen“. Ganz Links ist der Kreis in der Mitte das Quadrat und ganz Rechts das Dreieck. Traditionell stehen der Kreis und gebogene Linien für Feminität und Quadrate und Dreiecke für Maskulinität.[39] Kreise beziehen sich auf gebogene und geschwungene Linien. Quadrate sind verwandt mit geraden, vertikalen und horizontalen Linien. Dreiecke beziehen sich auf eckige und diagonale Linien. Objekte und Dinge, die Menschen als sicher ansehen, bestehen meist aus runden Formen und gebogenen Linien. Gefährliche Objekte oder Dinge bestehen meistens aus scharfen Linien und Dreiecken. In der Kunst reagieren Menschen auf Formen mit Erfahrungen aus dem echten Leben. [39]

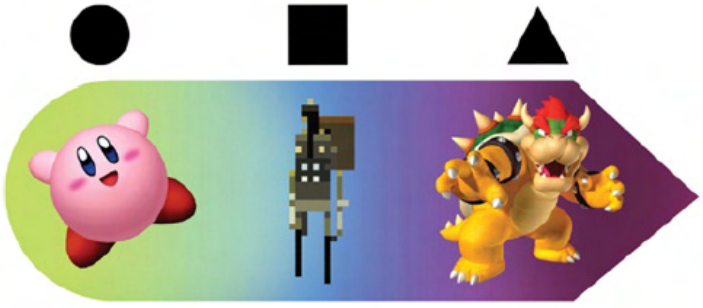


Abbildung 11.5: Skala der Formen[39]

11.2.2 Linien und Linien- Stile

Linien sind das einfachste Werkzeug für visuelle Kommunikation. Linien und Formen geben Konzepten, Werken und Bildern erst eine Form. Es gibt verschiedene Linienstile, mit denen der Erschaffende bestimmte Effekte erzeugen möchte. [39] [38]

Es ist möglich, in einem Bild Licht zu erschaffen, ohne etwas zu schattieren. Je nachdem Je nachdem wie viel Druck man einer Linie gibt, desto dunklere bzw hellere Konturlinien entstehen. Diese geben einer Oberfläche mehr oder weniger Licht. [39] [38]

Es gibt Linien, mit denen man in einem Bild oder in einem Werk Orientierung kommunizieren kann. Je nachdem in welchem Winkel eine Linie konstruiert wurde, kann sie die Orientierung oder Richtung eines Objekts bestimmen. Horizontal und vertikal gezeichnete Linien sind sehr statisch

und haben wenig Balance. Linien, die diagonal sind erzeugen eine bestimmte Illusion von Energie. [39] [38]

Man kann auch Tiefe in einem Werk erzeugen, indem man Linien in bestimmten Längen und Abständen aufträgt. Mit Linien kann auch Tiefe kreiert werden. Je nachdem, wie man die Stärke einer Linie ändert, erschafft das die Illusion von Atmosphäre und Perspektive. [39] [38]

Um eine starke Tiefe in Bildern zu erzeugen, werden oft Vorder -und Hintergrund Linien und Objekte übereinander gezeichnet. Das kreiert beim Betrachtenden das Gefühl von Tiefe in einer Komposition. [39] [38]

Um die Oberfläche eines Objekts zu beschreiben, ist auch die Form und Orientierung der Linie äußerst wichtig. [39] [38]

Linien, die sich in einem Werk entgegenkommen, erzeugen die Illusion von Bewegung und kreieren dadurch einen visuellen Pfad oder Weg für den Betrachtenden. Mit dieser Art von Linien kann man statische Objekte dynamisch wirken lassen. [39] [38]

11.2.3 Kompositionslinien

Kompositionslinien dienen zur Orientierung für den Betrachtenden. Sie setzen bewusst Blickpunkte, um das Auge zu lenken und den Schwerpunkt im Bild zu kommunizieren. Außerdem sollen Kompositionslinien Elemente im Bild miteinander verbinden, um Beziehungen zwischen Objekten und Bildelementen herzustellen. Kompositionslinien müssen für den Betrachtenden nicht immer sichtbar sein. [39] [38]

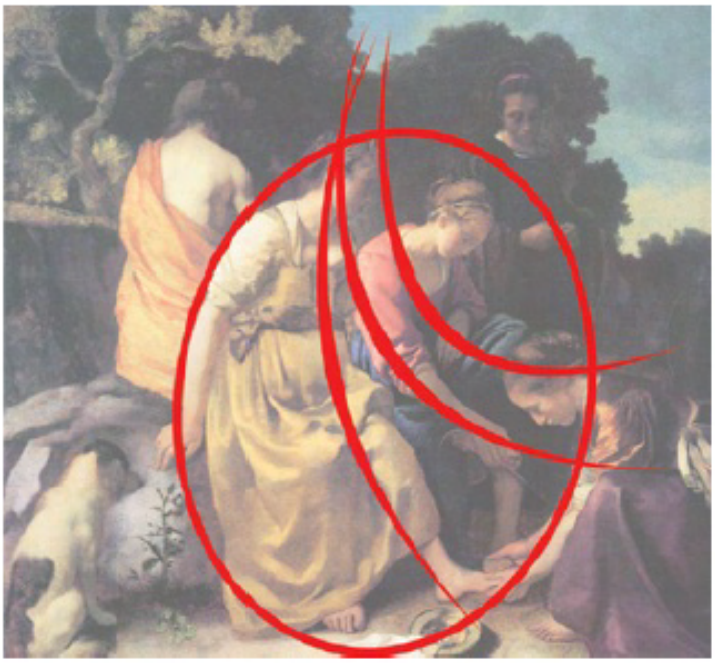


Abbildung 11.6: Kompositionslinien in einem Werk[39]

11.2.4 Wie Formen und Linien Emotionen erzeugen

Generell kann man sagen, dass jeder Form bestimmte Emotionen zugewiesen werden können. Jede Form hat eine andere Wirkung auf den Betrachtenden. Man kann so zwischen Gut und Böse unterscheiden, zwischen dynamischen oder statischen Objekten oder Charakteren und passivem und aggressivem Verhalten. Die drei primären Formen sind das Dreieck, der Kreis und das Quadrat. Der Kreis steht für Freundlichkeit und Positivität. Das Quadrat soll Vertrauen, Stabilität und Sicherheit kommunizieren. Das Dreieck steht für Gefahr und Aggressivität. Für Linien gilt dasselbe. Gebogen Linien wirken sehr harmonisch und ergänzend. Kantige, harte Linien sollen auf den Betrachtenden eher bedrohlich wirken. [39] [38]

11.3 Environment Design

Auch im Environment-Design sind Formen und Farben sehr wichtig, um beim Spielenden bestimmte Emotionen und Reaktion zu erzeugen. Die Formen und Farben die beim Designen gewählt werden haben einen sehr großen Einfluss auf das Erlebnis des Spielenden. [39]

11.3.1 Character und Environment Shapes

Welche Formen für Charaktere und Environment verwendet werden, hängt von dem Design des Spiels ab und von den Emotionen die man beim Spielenden erzeugen möchte. Die Beziehung vom Spielenden zum Charakter und dem Environment kann sich auch über die Zeit ändern, um bestimmte Emotionen im Prozess des Spieles zu erzeugen. [39] Um zwischen dem Charakter und dem Environment Harmonie zu kreieren, wird der Charakter aus Kreisen und geschwungenen Linien bestehen sowie das Environment. Man kann auch ein harmonisches Verhältnis zwischen dem Charakter und dem Environment erzeugen, indem der Charakter so gestaltet, dass er aus vielen Dreiecken besteht, genauso wie die Umgebung beziehungsweise das Environment. [39]

Um Unruhe und Dissonanz zwischen dem Charakter und der Umgebung zu schaffen, wird oft der Charakter des Spielenden aus vielen Kreisen gestaltet und das Environment aus vielen Dreiecken und kantigen Formen. Dissonanz kann auch andersrum geschaffen werden, der Charakter kann auch aus fast nur Dreiecken bestehen und die Umgebung aus Kreisen und weichen, geschwungenen Formen. [39]

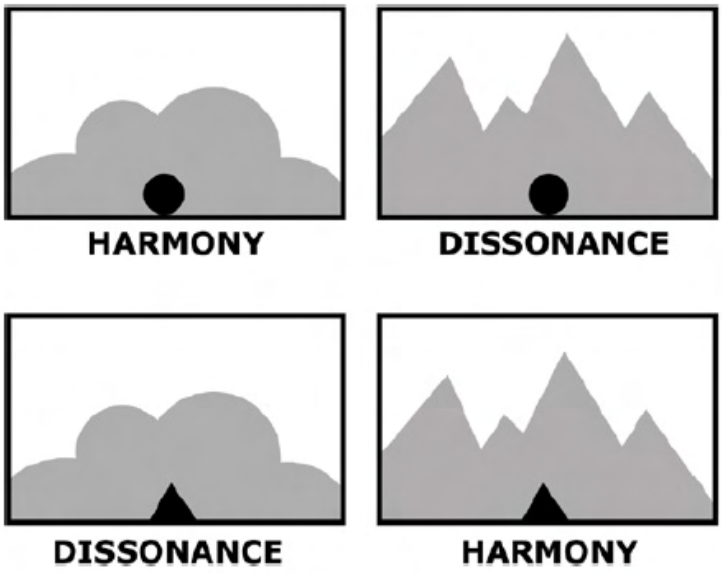


Abbildung 11.7: Character-Centric-Design[39]

11.3.2 Character Centric Environment Design

Beim Character-Centric-Environment-Design ist schon ein Charakter und eine Idee für die Umgebung vordefiniert. Es wird um den Charakter herum eine „Welt“, ein Environment kreiert. Je nachdem wie der Charakter

gestaltet wurde, welche Formen und Farben verwendet wurden und welche Emotionen man erzeugen möchte, wird die Umgebung um den Charakter gestaltet. Oft beeinflussen die Formen des Charakters die Gestaltung des Environments sehr. Es sollten bei dieser Art von Design immer zuerst die Charaktere und Bewohner der Umgebung gestaltet werden. [39]

Kapitel 12

Möglichkeiten zur Implementierung

12.1 Nutzung von Farben in Spielen

Die Nutzung von Farben in Videospielen war mit ihrer Entstehung sehr mit dem damaligen Stand der Technik verbunden. 1972 wurde die Farbüberlagerung erfunden. Diese hat es ermöglicht, dass Videospiele in Farbe dargestellt werden können. Davor konnten Spiele nur in Schwarz-Weiß oder Monochrom dargestellt werden. Ab Ende der 1980er sind fast alle Videospiele in Farbe. 1985 veröffentlichte „Sega“ das Sega Master System, das Spiele mit 32 verschiedenen Farben darstellen konnte. Schon 9 Jahre später veröffentlicht „Sony“ die Playstation, die es ermöglicht, Spiele mit rund 16.7 Millionen Farben darzustellen. Heutzutage können Spiele mit über 16.7 Milliarden verschiedenen Farben dargestellt werden. [44] [45]

12.1.1 Beispiel – Outlast

„Outlast“ ist ein Horror-Spiel, das 2013 für die Playstation 4, die Xbox One und den PC erschienen ist. Der Entwickler und Publisher ist „Red Barrels“. Der Journalist Miles Upshur erhält durch eine anonyme Quelle die Information, dass im Mount Massive Asylum unmenschliche Experimente an Patienten durchgeführt werden. Er geht dieser Spur nach und begibt sich auf den Weg zur Nervenklinik. Dort angekommen läuft alles ganz anders als erwartet. Es gibt keinen Weg nach draußen mehr und alle Insassen sind ausgebrochen. Ziel ist es, einen Weg aus der Nervenklinik zu finden und währenddessen alles mit einem Camcorder zu filmen. [46] [45]

Abbildung 12.1: “Outlast“ In-Game Screenshot[39]

In „Outlast“ wird die Atmosphäre und Stimmung des Spiels durch den Qualitätskontrast der Farben erzeugt. Der Qualitätskontrast wird durch die Farbqualität der Farbe erzeugt. Durch das Mischen mit Schwarz, Weiß oder Grau wird eine Farbe unrein. Die Farbe verliert als Folge an Leuchtkraft und Qualität. Farben, die mit Grau gemischt werden, wirken trüb. Wird eine Farbe mit Schwarz gemischt, so wirkt sie bedrohlicher. Wenn man eine Farbe mit Weiß mischt, so wird sie kälter. Wendet man einen Qualitätskontrast neben reinen Farben an, fallen sie stärker auf. [47] [46] [45]

In „Outlast“ wird dem Spielenden durch den Qualitätskontrast eine stark bedrohliche Umgebung vermittelt. Alles ist sehr düster und dunkel, erst wenn der Spielende den Camcorder benutzt, wird die Umgebung in ein blasses Grün getaucht. Die Farben wirken größtenteils sehr leblos, aber gleichzeitig bedrohlich. Die einzig gesättigte, reine Farbe ist Rot in Form von Blut überall in der Anstalt. [46] [47] [45]

12.1.2 Beispiel – Journey

„Journey“ ist ein Adventure-Spiel, das ohne jegliche Worte auskommt. Es wurde erstmals 2012 für die Playstation 3 veröffentlicht. Der Entwickler des Spiels ist „thatgamecompany“ und der Publisher „Sony Interactive Entertainment“ und „Annapurna Interactive“. Das Spiel startet in einer Wüste. In der Ferne liegt ein großer Berg, an dessen Gipfel ein Lichtstrahl in den Himmel ragt. Der Spielende spielt eine Figur in einer roten Robe. Auf dem Weg zum Berg kann der Spielende mit bestehender Internetverbindung auf einen anderen Spielenden treffen. Die Spielenden können nicht miteinander reden, aber sich dennoch gegenseitig helfen. Das Zeil von „Journey“ ist es, den in der Ferne liegenden Berg zu erreichen. [39] [45]

„Journey“ nutzt die Vorteile von Farbkontrasten, um bei den Spielenden

Emotionen zu erzeugen. Das Spiel bedient sich am Warm-Kalt-Kontrast. Beim Warm-Kalt-Kontrast werden warme und kalte Farben einander gegenübergestellt. Warme und kalte Farben stehen sich im Farbkreis gegenüber und erzeugen deshalb Dissonanz. Die kalten Farben des Farbkreises sind Gelbgrün, Grün, Blaugrün, Blau, Blauviolett und Violett. Zu den warmen Farben des Farbkreises zählen Gelb, gelborange, Orange, Rotorange, Rot und Rotviolett. [45] [39] [47]

Abbildung 12.2: "Journey" In-Game Screenshot[39]

In „Journey" wechseln sich die Warm-Kalt-Kontraste immer wieder ab. Am Anfand des Spiels ist der Spielende in einer Wüste, nach der Wüste führt ihn seine Reise in eine kühle Höhle. Nach den warmen, sandigen Ruinen begibt sich der Spielende zum eiskalten Berg in der Ferne. Die Figur selbst trägt eine rote Robe die in der kühlen Höhle und am kalten Berg den Warm-Kalt-Kontrast nochmal verdeutlicht. [39] [45] [47]

12.2 Nutzung von Formen in Spielen

Das Konzept und die Nutzung von Formen und Linien in Videospielen stammt aus der Natur. Objekte und Dinge die aus abgerundeten Formen bestehen empfinden Menschen als sicher. Kantige Objekte und Formen gelten als gefährlich. Der Kreis steht für Freundlichkeit und Positivität. Das Quadrat soll Vertrauen, Stabilität und Sicherheit kommunizieren. Das Dreieck steht für Gefahr und Aggressivität. Man kann diese Formen in einer Skala der Emotionen einordnen. Links ist die positivste Form, der Kreis und ganz rechts das Dreieck, die negativste Form. [39]

In Videospielen bestehen die „guten" Charaktere beziehungsweise die Helden aus vielen Kreisen und abgerundeten Linien. Oft enthalten ihre Designs auch Quadrate, um Sicherheit und Stabilität zu signalisieren. Die Ant-

agonisten in Spielen haben meist sehr dominante Silhouetten, die aus Dreiecken oder sehr kantigen Linien bestehen. Weiters, ist es äußerst wichtig, Silhouetten aus großen Formen zu gestalten, um die Figur des Charakters herausstechen zu lassen. [39]

Abbildung 12.3: Video-Game Character-Shapes[39]

In Abbildung 12.3 kann man erkennen, dass in der linken Spalte die Protagonisten aus sehr abgerundeten Linien und Kreisen bestehen. Sie wirken auf den Betrachtenden sehr positiv. Auf der rechten Seite sind die Antagonisten, die aus Dreiecken und vielen kantigen Linien bestehen. Sie wirken auf den Betrachtenden sehr aggressiv und bedrohlich. Generell sind alle abgebildeten Charaktere aus markanten Formen gestaltet, damit sie möglichst gut herausstechen. [39]

12.2.1 Beispiel – Journey

In „Journey“ wird auch eine positive, harmonische Stimmung durch die Nutzung von Dreiecken und kantigen Linien erzeugt. Der Charakter besteht aus einem Dreieck und sein Umfeld ebenso. Die Umgebung ist geprägt durch kantige Objekte und Formen und dreieckige Figuren. Durch die Nutzung von Dreiecken für Charakter und Umfeld wird Harmonie erzeugt. [39] [45]

Abbildung 12.4: “Journey“ In-Game Screenshot [48]

12.2.2 Beispiel – Super Mario Galaxy

„Super Mario Galaxy“ ist ein 3D-Plattformer, der 2007 für die Wii veröffentlicht wurde. Der Entwickler und Publisher des Spiels ist „Nintendo“. Der Protagonist Mario reist von Planet zu Planet, kämpft gegen Gegner, löst Rätsel und sammelt Items. Ziel des Spiels ist es, das Pilzkönigreich und Prinzessin Peach zu retten. [49] [39]

Mit Formen und Linien kann man bei den Spielenden Gefühle und Emotionen erzeugen. Je nachdem, welche Emotion beim Spielenden erzeugt werden soll, werden andere Charakter -und Environment-Shapes verwendet. Um Harmonie in einer Umgebung zu erzeugen, besteht der Charakter und sein Umfeld aus Kreisen und runden Linien. Harmonie kann aber auch durch Dreiecke und kantige Linien entstehen. Um Dissonanz zu erzeugen, besteht der Charakter aus Kreisen und gebogenen Linien oder Dreiecken und kantigen Linien und das Umfeld steht im Gegensatz dazu. [39]

Abbildung 12.5: “Super Mario Galaxy“ In-Game Screenshot[39]

In „Super Mario Galaxy“ besteht auf den meisten Planeten Harmonie zwischen dem Charakter und seiner Umgebung. Mario besteht größtenteils aus Kreisen und runden, gebogenen Linien. Seine Umgebung besteht im Großen und Ganzen auch aus Kreisen und abgerundeten Linien. Diese Beziehung zwischen Charakter und Umgebung erzeugt Harmonie und löst bei den Spielenden positive Gefühle aus. [39]

Kapitel 13

Konkrete Implementierung

Das Ziel ist es eine harmonische Atmosphäre zwischen den Charakteren und der Umgebung zu schaffen. Einerseits gibt es die Umgebung der PC-Spielenden und andererseits die Umgebung des VR-Spielenden. Die Snowboarder sind niedliche, freundliche Tiere die dem Betrachtenden ein positives Gefühl vermitteln sollen. Der Magier ist der Antagonist des Spiels und soll somit einen eher gefährlicheren aber auch mystischen und magischen Eindruck hinterlassen. Die Atmosphäre des Umfelds des VR-Spielenden und der PC-Spielenden soll harmonisch mit den Designs der Charaktere wirken. Das Environment der PC-Spielenden soll sehr niedlich, positiv und freundlich auf die Betrachtenden wirken. Es soll das Thema einer winterlichen Schneelandschaft gemischt mit Süßigkeiten gut widerspiegeln. Die Umgebung des VR-Spielenden soll eher eine düstere, mysteriöse Stimmung haben. Sie soll aber auch das Überthema „Süßigkeiten“ wieder aufgreifen. Das Hauptmenü der PC-Spielenden soll ebenfalls das süße, niedliche und positive Thema aufgreifen, um ein harmonisches Gesamtbild zu ergeben.

13.1 Konzeption der Charaktere

In „Tricks ‘n‘ Treats“ gibt es verschiedene Charaktere, die dem PC-Spielenden zur Verfügung stehen. Der VR-Spielende hat nur einen Charakter zur Verfügung. Es gibt vier verschiedene Tiere, einen Hasen, einen Fuchs, einen Eisbär und einen Wolf, zwischen denen die PC-Spielenden wählen können. Der VR-Spielende spielt einen Magier. Die Charaktere wurden so konzipiert, um in der jeweiligen Umgebung harmonisch zu wirken und um dem Spiel eine atmosphärische Wirkung zu geben. Generell sollen die Tiere sehr niedlich, süß und freundlich auf die Spielenden wirken. Der Magier soll eine mystische, geheimnisvolle, aber auch bedrohliche Stimmung erzeugen. Die verwendeten Farben und Formen sollen die Gefühle, die den Spielenden vermittelt werden sollen, verstärken.

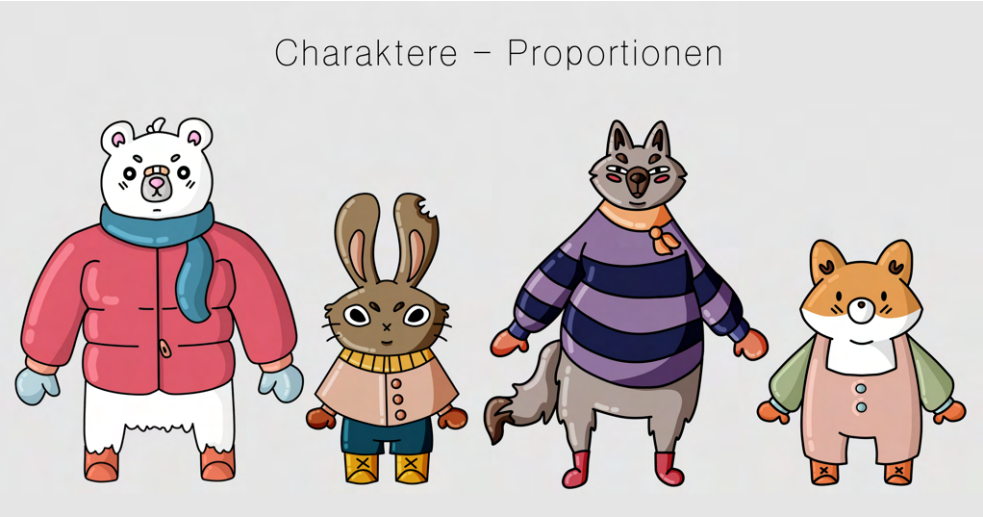


Abbildung 13.1: Character-Design der Snowboarder

Der Eisbär ist einer der vier Charaktere, die den Spielenden zur Auswahl stehen. Die Hauptfarben des Bären sind Weiß, Pink, Orange und ein helles und etwas dunkleres Blau. Obwohl der Eisbär außerhalb des Spiels ein Raubtier ist, wirkt sein Design sehr niedlich und freundlich. Vor allem die Farben Pink und Weiß bekräftigen diese Gefühle stark. Die Farbe Pink wirkt auf den Betrachter äußerst beruhigend. Weiß steht für Reinheit und Unschuld. Außerdem sticht das Pink der Jacke des Bären in der winterlichen Umgebung stark heraus. Da der Bär ein größeres, sehr stabiles Tier ist, besteht er überwiegend aus Quadraten. Quadrate signalisieren Stabilität und Sicherheit. Der Eisbär soll aber auch sehr niedlich und süß wirken und besteht aufgrund dessen auch aus einigen Kreisen. Kreise signalisieren Freundlichkeit und Positivität. Durch die Verwendung von Quadraten und Kreisen wirkt der Charakter sehr sicher und stabil, aber auch niedlich und süß.

Der Hase ist ebenfalls einer der vier Charaktere, die den Spielenden zur Verfügung stehen. Die Hauptfarben des Tiers sind Hellbraun, Gelb, Rosa und ein dunkles Blau. Der Hase soll wie die anderen Charaktere eine positive Ausstrahlung auf den Spielenden haben. Das signalisiert vor allem die Farbe Gelb, Gelb wird mit der Sonne assoziiert und mit glücklichen Emotionen. Auch Rosa hat auf den Betrachtenden eine freundliche, positive Wirkung. Auch hier heben sich die Farben des Gewandes und des Hasen von der winterlichen Schneelandschaft ab. Der Hase wirkt nicht nur durch seine Farben sehr freundlich und süß, sondern auch durch die verwendeten Formen. Er besteht größtenteils aus Kreisen und abgerundeten Linien, dadurch vermittelt er dem Betrachtenden ein sicheres, gutes Gefühl, er wirkt sehr harmlos und ungefährlich. Da die Tiere Snowboarder sind und einen Sport

ausüben, soll er auch stabil wirken, das wird durch einen geringen Anteil an Quadraten in seinem Design erreicht.

Der Wolf ist einer der vier spielbaren Charaktere. Die Hauptfarben dieses Tiers sind Orange, Violett und ein wärmeres Grau. Da Orange für Kreativität und Enthusiasmus steht, ist sie auch eine Farbe, die mit positiven Emotionen verbunden wird. Außerdem wird sie vom menschlichen Auge als wärmste Farbe im Farbkreis wahrgenommen. Die graue Fellfarbe des Wolfes ist auch mit einem warmen Farbton gemischt und wirkt deshalb freundlich auf den Betrachtenden. Generell heben sich die drei verschiedenen Hauptfarben des Wolfes gut von der Schneelandschaft und seiner Umgebung ab. Die Figur des Wolfes besteht aus vielen Kreisen und abgerundeten Linien, das macht ihn zu einem sehr freundlichen, eher harmlosen Charakter. Sein Design beinhaltet aber auch einige Quadrate und Vierecke, um zu symbolisieren, dass er eine sehr stabile Statur hat, denn der Wolf ist einer der beiden größeren spielbaren Charaktere.

Der letzte spielbare Snowboarder ist der Fuchs. Er ist wie der Hase einer der beiden kleineren Figuren. Die Hauptfarben des Fuchses sind ein helles und dunkleres Orange, ein helles Grün und ein heller rosafarbener Farbton. Grün ist für das menschliche Auge die erholsamste Farbe und wird mit der Natur verbunden. Orange steht für Kreativität und Enthusiasmus, weshalb sie eine sehr positive Wirkung auf den Betrachtenden hat. Auch Rosa hat eine beruhigende Wirkung und verursacht beim Spielenden gute Emotionen. Durch diese Kombination von Farben und Farbtönen wirkt der Fuchs sehr niedlich und freundlich auf den Spielenden. Weiters heben sich seine Hauptfarben auch sehr gut von der winterlichen Umgebung ab. Der Fuchs besteht auch aus sehr vielen abgerundeten Formen und Linien. Sein Design basiert auf dem Kreis und dem Quadrat, welche beim Betrachtenden positive Gefühle erzeugen. Die abgerundeten Formen lassen den Charakter sehr niedlich, süß und ungefährlich wirken.

Der VR-Spielende steuert den Magier, der Magier ist der Antagonist des Spiels. Seine Farben wurden so gewählt, um seine Rolle als Rivale der Snowboarder zu verstärken. Seine Hauptfarben sind ein sehr dunkles Grau, Violett und Pink. Dunkles Grau wird genauso wie Schwarz mit eher schlechten Emotionen verbunden. Schwarz steht für Tod, Angst und Negativität. Die Farbe Violett steht für Magie und Mystik, weshalb sie sehr gut zum Charakter passt. Der Magier besteht größtenteils aus Dreiecken und härteren Linien. Dreiecke wirken auf den Betrachtenden aggressiv und signalisieren Gefahr. Da der Charakter des VR-Spielenden der Antagonist des Spiels ist, wurde er auch formlich so gestaltet. Besonders die dreieckigen, spitzen Augen lassen ihn besonders gefährlich erscheinen.

Abbildung 13.2: Artwork des Magiers

13.2 UI/UX Design

Das „User-Interface“ beschränkt sich größtenteils auf das Erlebnis der PC-Spielenden. Aus diesem Grund wurde das UI passend zu den Snowboardern und dem Environment der Snowboarder gestaltet, um ein harmonisches Gesamtbild zu kreieren. Die Atmosphäre soll eine sehr positive, süße und niedliche sein, die beim Betrachtenden gute Emotionen und Gefühle erzeugt. Das UI soll diese Thematik widerspiegeln.

Im Hauptmenü wurden die Buttons mit einem pinken Farbton versehen. Die Farbe Pink hat einen beruhigenden Effekt für den Betrachtenden und löst glückliche Emotionen aus. Die Farbe wurde auch passend zu dem poppigen Überthema „Süßigkeiten“ gewählt. Die Buttons sollen auch in einem harmonischen Zusammenhang mit den Hintergrundbildern des Hauptmenüs stehen. Außerdem hebt sich die Farbe sehr gut von den Hintergründen ab und dadurch entsteht ein Kontrast, der sich stark auf das UI fokussiert.

Abbildung 13.3: Menu-Screen-Mockup

Das Design des Hauptmenüs wird auch in allen anderen Menüs wieder aufgegriffen. Die abgerundeten Ecken und runden Formen sowie die knalligen Farben ziehen sich durch das ganze User-Interface des Spiels. Die UI-Elemente wurden immer mit dunkleren Linien umrandet, um einen Kontrast zwischen dem UI-Element und dem Hintergrund zu schaffen. Dadurch ist das User-Interface für die Spielenden klar erkennbar beziehungsweise gut sichtbar. Das „Selection-Menu“ wurde mit diesen gestalterischen Mitteln kreiert. Es ist ein gutes Beispiel für ein harmonisches Gesamtbild zwischen UI-Elementen, Hintergrund und anderen Bildelementen. Die Farben der UI-Elemente stechen in der Komposition am meisten heraus und sind somit klar für die Spielenden erkennbar.

Abbildung 13.4: Selection-Menu-Mockup

Auch das In-Game UI der PC-Spielenden wurde passend zu der Umgebung und der Charaktere gestaltet. Die Hauptfarben des User-Interface sind Pink und Gelb. Sie spiegeln die fröhliche Stimmung wieder und sehr hervor, damit sie für die Spielenden gut erkennbar sind. Weiters, sind die Icons noch mit weißen Linien umrandet, damit sie deutlich erkennbar sind. Es sind außerdem alle Elemente mit abgerundeten Linien und Kreisen gestaltet worden, um ein harmonisches Gesamtbild zu erzeugen.

13.3 Environment Design und Komposition

Die Umgebung der Spielenden wurde farblich und formlich gestalterisch an ihre Charaktere angepasst. Es wurden insgesamt zwei verschiedene Umgebungen gestaltet, die der PC-Spielenden und jene des VR-Spielenden.

Für das Environment der PC-Spielenden wurden sehr knallige, poppige Farben verwendet, die sich gut von der weiß-blauen, schneebedeckten Strecke abheben. Dadurch können die Spielenden Hindernisse und Objekte auf der Strecke gut voneinander unterscheiden. Durch diese Farbkonzeption ist es klar geregelt wo es für die Spielenden sicher ist zu fahren und wo nicht. Weiters, sind auch alle Assets formlich passend zu den Charakteren gestaltet worden. Sie sind alle sehr rundlich und ergeben so ein harmonisches Gesamtbild.

Abbildung 13.5: Environment-Konzept

Es wurde für den Start-Screen des Spiels ein Hintergrundbild gestaltet, das das Environment und die Charaktere noch mal in einem harmonischen Gesamtbild zeigt. Die Charaktere heben sich klar vom Hintergrund ab, nicht nur durch ihre poppigen Farben, sondern auch durch ihre Silhouetten. Alle Vordergrundelemente wurden mit sehr knalligen Farben versehen, um einen Kontrast zwischen Vorder -und Hintergrund entstehen zu lassen. Durch diese farbliche Trennung entsteht auch Tiefe in der Bildkomposition. Außerdem wurden bei der Gestaltung Kompositionslinien verwendet, die das Auge des Betrachtenden auf die wichtigsten Bildelemente fokussiert. Durch die Verwendung von Kompositionslinien wurde das Bild sehr dynamisch.

Abbildung 13.6: Start-Menu-Screen

Die Umgebung des VR-Spielenden wurde passend zu dem Design des Magiers gestaltet. Atmosphärisch ist alles recht düster und dunkel, dennoch wurde versucht, es etwas farbenfroher darzustellen. Die Farben des Magiers finden sich in seiner Umgebung immer wieder, um ein harmonisches Gesamtbild zu kreieren. Generell ist der Raum des VR-Spielenden im Gegensatz zu dem Environment der PC-Spielenden sehr kantig und eckig. Es finden sich viele harte Formen und Dreiecke. Von der formlichen Gestaltung stehen so Charakter und Umgebung in einer harmonischen Beziehung zueinander.

13.4 Andere 2D-Assets

Weitere nennenswerte Assets wären das Comic für die Start-Sequenz sowie Texturen für 3D-Assets und jegliche Icons und Hintergründe. Alle anderen Assets, die in den oberen Absätzen nicht genannt wurden, wurden farblich und formlich an den Stil des Spiels angepasst. Viel Arbeit wurde in den Comic investiert, der die Vorgeschichte des Spiels erzählt. Von den Farben wurde er passend zum Gesamtbild von „Tricks ‘n‘ Treats“ gestaltet. Er geht auch weiter auf die Persönlichkeiten der einzelnen Charaktere ein.

13.5 Fazit

Im Großen und Ganzen gab es bei der 2D-Asset Produktion keine großen Schwierigkeiten. Die Produktion der Assets verlief wie geplant und die Endergebnisse sind geworden, wie sie angedacht wurden. Von der Konzeption bis hin zur Umsetzung verlief die Ausarbeitung sehr gut. Würde an dem Projekt in Zukunft weitergearbeitet werden, würde das Projekt um neue Strecken

erweitert werden. In Folge würde das bedeuten, dass mehr Environment-Konzepte nötig wären und viele weitere 2D-Assets produziert werden müssten. Von User-Interface Elementen bis hin zu Hintergründen und Texturen. Es könnten auch neue Charaktere implementiert werden, was zur Folge hätte, dass die Story-Line des Spiels erweitert werden müsste. Außerdem müssten neue Comics gezeichnet werden, um die Geschichte bildlich darzustellen. Das Projekt war für die persönliche Weiterentwicklung ein wichtiger Schritt, da das Gefühl für Farben und Formen bzw. Farb -und Formpsychologie stark vertieft wurde. Nicht nur das Gefühl für Farben und Formen wurde vertieft, sondern auch die Fähigkeit in einem Team zu arbeiten. Die Teamarbeit in unserem Projekt funktionierte sehr gut und es gab wenig Probleme. Es scheiterte nur selten an Kommunikationsschwierigkeiten.

Performance-Optimierung von 3D Assets in VR-Spielen

Ian Memic

DIPLOMARBEIT

eingereicht am
Ausbildungsschwerpunkt
Mediendesign - Gamedesign

betreut durch
Alexander Hager

im März 2023

Kapitel 14

Überblick über die Materie

14.1 Performance Optimierung

Die Performance Optimierung lässt sich allgemein als der Prozess der Verbesserung einer getätigten Leistung beschreiben. Mit dieser Definition lässt sich ein schnellerer Weg von Punkt A zu Punkt B als eine Weg-Zeit Optimierung darstellen. In der Computer Technik beschreibt die Performance Optimierung daher grundsätzlich, dass ein Computer System oder eine Computer Software in bestimmten Bereichen effektiver bzw. mit weniger Rechenleistung läuft. Für Computerspiele ist die Performance Optimierung ein essenzieller Teil der Spiele-Produktion vor der Veröffentlichung. Code, Game-Physics, Game-Assets und allgemeine Grafik Optimierung bilden damit den Überbegriff: Performance Optimierung. Dies erklärt dennoch nicht deutlich genug, was genau die Optimierung von 3D-Assets bedeutet und wird im Folgenden konkret erklärt.[50, S. 648]

14.1.1 Was ist schlechte Performance?

Schlechte Performance ist eindeutig erkennbar und kann in Spielen in zahlreichen Formen auftreten. Die gewöhnlichsten Formen davon sind eine niedrige Framerate, Standbilder, lange Ladezeiten, Abstürze und Fehlermeldungen, Grafikfehler, inkonsistente In-Game-Physik, falsches Benehmen von Spiel-Ereignissen und auch hoher Strom verbrauch.

14.1.2 Was löst schlechte Performance aus?

Damit das Spiel auf einem Computer laufen kann, muss der Computer Arbeit verrichten, welche Rechenleistung benötigt. Die CPU (= Central-Processing-Unit) und GPU (= Graphics Processing Unit) benötigen Ressourcen, um Prozesse auf dem Computer laufen zu lassen. Genauso ist auch der RAM (=Random-Access-Memory) zu beachten. Die Überlastung dieser Komponenten führt daher zu verschlechterter Performance. Es ist wichtig

anzumerken, dass die Überlastung sowohl durch die Entwickler als auch durch die Nutzer entstehen kann.[51, S. 17] Spiele können nur bis zu einem gewissen Grad auf eine bestimmte Mindestanforderung optimiert werden, wenn jedoch der Computer zu einer veralteten Generation angehört, wird jegliche Optimierung sinnlos sein. Die Entwickler sind zuständig für alle ausgeführten Prozesse während der Laufzeit des Spiels und deswegen in vielen Instanzen hauptverantwortlich für ein schlecht laufendes Spiel.

14.2 Rendering & die VR Kamera

14.2.1 Rendering

Rendering ist ein Fachausdruck aus der Computer-Grafik. Rendering umschreibt jenen technischen Prozess, mit welchem die GPU und CPU durch Rohdaten ein Bild auf einem Bildschirm generiert. Rendering ist daher Grund, weshalb unser Computer während der gesamten Laufzeit ein Bild darstellen kann. Daraus folgt, dass jedes 3D-Assets und ebenso jedes Spiel mittels Rendering ein Bild anzeigt. In Spielen und 3D-Modeling wird zwischen zwei wichtigen Arten von Rendering unterschieden: Real-Time Rendering: Hier handelt es sich um alle Rendering-Prozesse in Echtzeit (Realtime) bzw. während der Laufzeit berechnet werden und deswegen für interaktive Ereignisse nützlich sind.[52] Offline-Rendering ist das Gegenstück zum Real-Time-Rendering. Hierbei wird ein Bild vorzeitig fertig berechnet und erspart während der Laufzeit einiges an Leistung.[52] Wie bereits erklärt, kann ein Spiel ohne Rendering kein Bild ausgeben. Damit ein Bild ausgegeben werden kann, wird in Game-Engines wie die Unity-Engine, eine virtuelle Kamera benötigt. Diese virtuelle Kamera ist schlussendlich die Perspektive, aus welcher Nutzer das Spiel erleben werden. Die Perspektive kann von Spiel zu Spiel variieren und bestimmt die Stimmung und das Erlebnis des Spieles. Der tatsächliche Rendering-Prozess ist dabei nicht abhängig von der Kamera und bleibt in den meisten Umständen gleich. Es ist wichtig anzumerken, dass die Anzahl der aktiven Kameras in der Game-Szene die Anzahl an einzelnen Rendering-Prozessen erhöht. Jede aktive Kamera muss mittels CPU und GPU gerendert werden und erhöht dabei die Rechenlast. Als Beispiel nehmen wir eine Unity-Scene, in welcher sich drei Objekte und zwei Kameras befinden, die aus unterschiedlichen Winkeln auf die Objekte zeigen. Obwohl es sich immer noch um drei Gegenstände in jeder Kamera handelt, müssen alle Objekte aus jeder Kamera gerendert werden, weswegen nun 6 Objekte gerendert werden müssen.

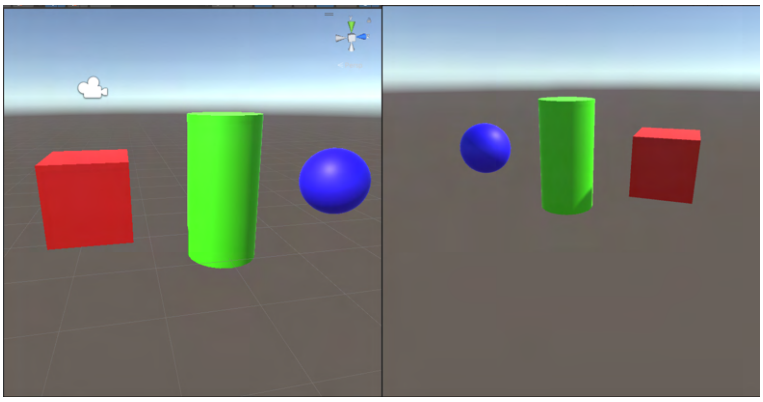


Abbildung 14.1: Rendering Beispiel: 2 Kameras in Unity

14.2.2 VR-Kamera

Der Zweck von Virtual Reality ist es, einen möglichst immersiven Raum darzustellen. Die VR-Brille besitzt zwei Linsen, durch welche der/die Träger/in das virtuelle Bild wahrnehmen. Die VR-Kamera muss dafür genauso rendern. Damit es sich jedoch nicht um ein statisches zweidimensionales Bild handelt, ist es unentbehrlich, dass die Szene mittels zwei Kameras aus unterschiedlichen Winkeln aufgenommen wird. Die VR-Linsen simulieren damit die Augen eines Menschen, um das Gefühl eines wirklichen Raumes zu realisieren. Pro Linse wird jeweils eine Kamera benötigt, woraus sich erhöhte Rendering-Kosten ergeben. Eine Unity-Scene mit integriertem VR-Headset muss deswegen die gesamte Szene doppelt rendern. VR-Technologie stellt damit ein großes Performance-Bottleneck für Entwickler dar, weswegen sich der Markt nur langsam verbreitet. Realistische oder detaillierte Szenen sind mit effektiver Optimierung der 3D-Assets dennoch möglich.

Abbildung 14.2: VR-Sicht mit zwei Linsen

[53]

14.3 Mesh Optimierung

Game-Assets lassen sich mit den Requisiten in Filmen vergleichen. Assets beziehen sich auf jene digitalen bzw. virtuellen Gegenstände, welche für die Spielentwicklung benötigt werden. Hierzu gelten 2D Art und 3D-Models, Animationen oder Audiofiles. Die 3D-Assets werden von 3D-Artists erstellt und in das Spiel importiert. Hier kann es sich um jeden möglichen Gegenstand, wie bspw. eine Tasse oder ein Charakter handeln.[51]

14.3.1 Das Mesh

3D-Assets auch Mesh Objekt ist ein vom Computer generiertes Polygonnetz. Ein Polygon setzt sich aus mehreren Polytopen (Punkten) und den damit entstehenden Streckenzug (Kanten) zusammen, welche dann das Vieleck bilden. Beim Polygonnetz handelt es sich somit um eine Anreihung zahlreicher Polygone nebeneinander, welche gemeinsam die Geometrie des Mesh bilden. Ein Polygon muss aus mindestens drei Polytopen bestehen, um eine geschlossene Fläche zu bilden. Damit stellt das Dreieck die einfachste Variante eines Polygons dar.[54]

14.3.2 Geometrie vs. Topologie

Ein Mesh kann auf zwei verschiedene Arten betrachtet werden: Geometrie und Topologie. Diese zwei Begriffe werden oft verwechselt, doch beschreiben sie andere Eigenschaften des Meshes. Die Geometrie eines Meshes beschreibt den Körper als Gesamten. Das bedeutet, dass die Geometrie die Form und Maße des Objektes umfasst. Bei der Topologie spricht man von der Struktur

eines Meshes. Ein Mesh kann bspw. zweimal in derselben Form existieren, würde sich aber in seiner Struktur unterscheiden. Diese Struktur bezieht sich dabei auf die Anordnung seiner Vertices, Edges und Faces und wie die Komponenten miteinander verbunden sind.

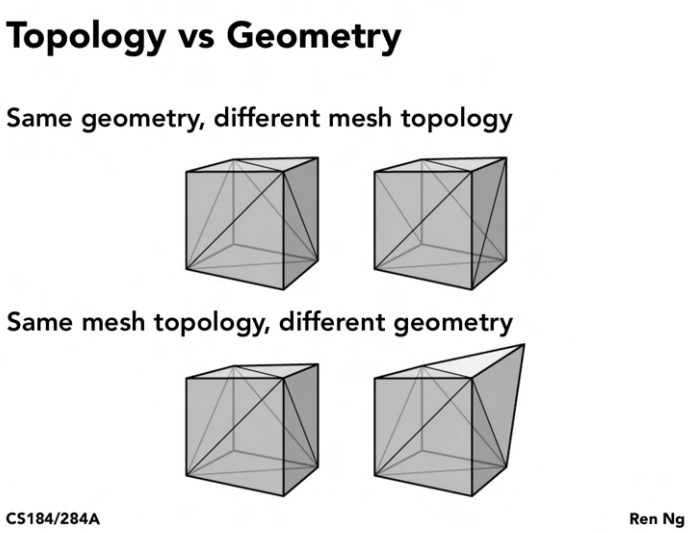


Abbildung 14.3: Unterschied zwischen Topologie und Geometrie[55]

In Essenz besteht das Mesh aus 3 wichtigen Komponenten[56, S. 2]:

Vertex (Mehrzahl Vertices)

Ein einzelner Punkt (= Polytop) Sie stellen einen Punkt dar und werden mittels Vektors beschrieben. Damit können die Koordinaten des Punkts in einem Koordinatensystem angegeben werden.[54, S.47]

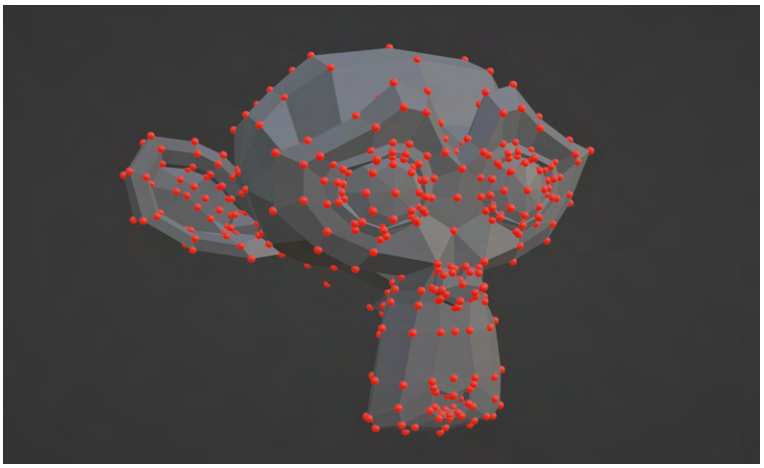


Abbildung 14.4: Visuelle Darstellung von Vertices

Edge

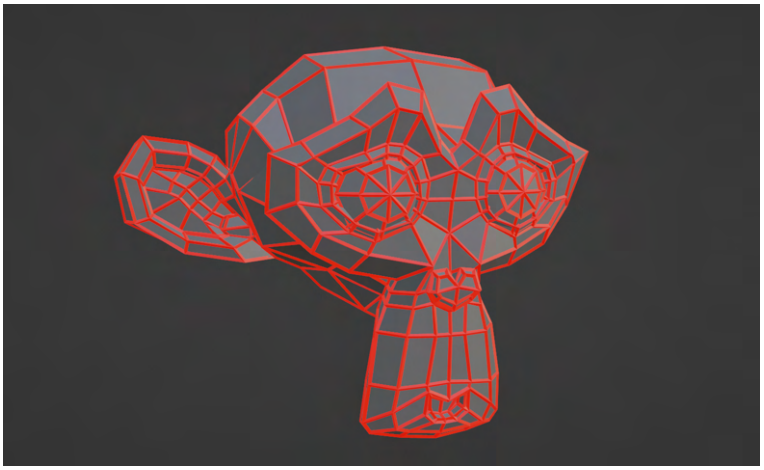


Abbildung 14.5: Visuelle Darstellung von Edges

Es entsteht aus 2 Punkten und bildet damit eine Linie/Kante

Face (Polygon)

Es besteht aus mindestens 3 verbunden Edges, die eine Fläche bilden.

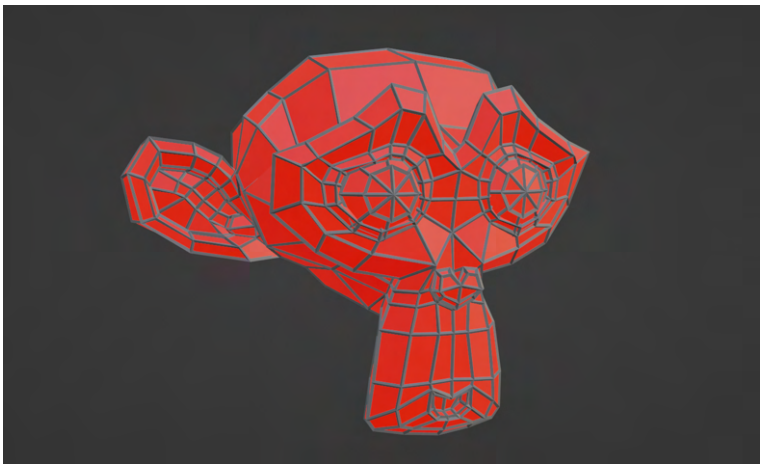


Abbildung 14.6: Visuelle Darstellung von Faces

14.3.3 Tris, Quads und N-Gone

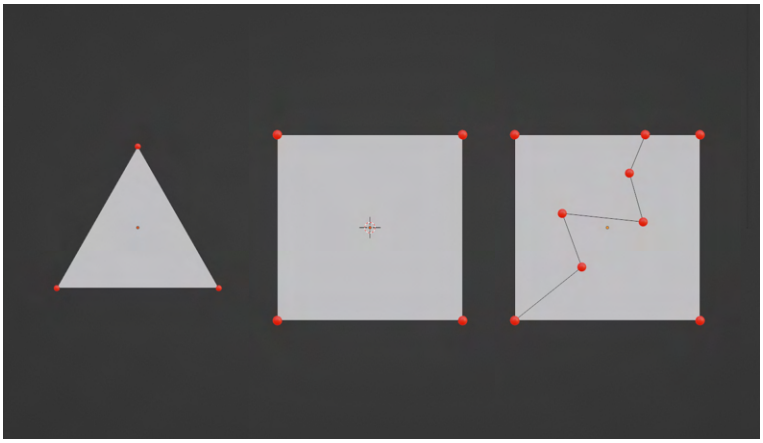


Abbildung 14.7: Tri, Quad & N-Gon

Streng genommen gibt es kein Maximum an Edges und Vertices, die ein Face besitzen kann. Jedoch sind zwei Arten von Polygonen besser geeignet als andere. Hierbei spricht man fachlich von Quads (Vierecken) und Tris (Dreiecken). Überschreitet ein Face die Anzahl an Vertices eines Quads, also vier Vertices, spricht man von einem N-Gon. N-Gons unterscheiden sich vom Rendering-Prozess nicht von allen anderen Polygonen, jedoch führen N-Gons meist zu irritierenden Rendering-Ergebnissen und anderen grafischen Fehlern. Weiterführend sind die meisten 3D-Softwares nicht darauf ausgelegt, mit N-Gons zu arbeiten, da der Workflow in 3D-Modeling Programmen auf optimierte Topologie basiert. Bevor ein Mesh exportiert wird, wird

die Topologie des Mesh meist aus Quads bestehen, da das Model bei notwendigen Veränderungen leichter bearbeitbar ist. Sobald das Asset in die Game-Engine importiert wurde, konvertiert die Engine jegliche Quads zu Tris. Dies liegt daran, dass jedes Quad in zwei Tris aufgeteilt werden kann. Der Vorteil beim Rendering dabei ist, das die Fläche eines Dreiecks planar ist, im Gegenzug zu dem eines Quads. Dies bietet den Vorteil, dass nur die einfachste geometrische Fläche gerendert werden muss und damit auch den Vorteil, dass keine Probleme beim Shading des Models vorkommen.

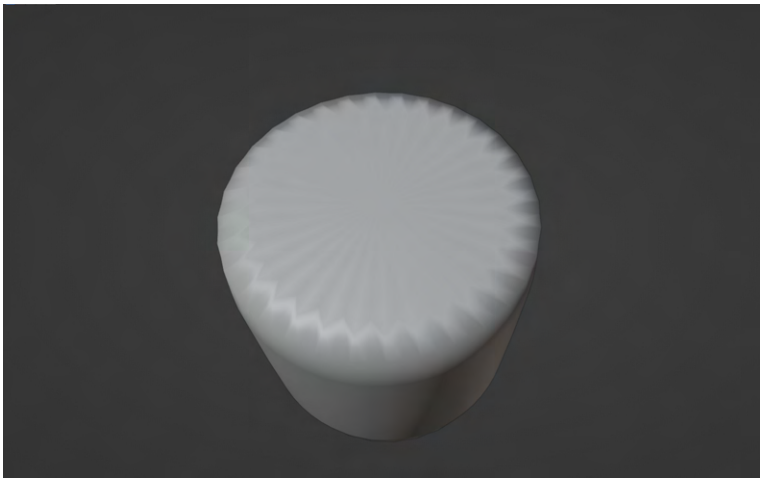


Abbildung 14.8: Shading-Probleme bei N-Gons

14.3.4 Die Wichtigkeit von Performance Optimierung in Video Spielen

Dr Davide Aversa und Chris Dickinson, erläuterten die Wichtigkeit der Performance folglich:

„The goals of performance optimization are deeply entwined with user experience. Poorly optimized games can result in low frame rates, freezes, crashes, input lag, long loading times, inconsistent and jittery runtime behaviour, physics engine breakdowns, and even excessively high battery power consumption (an often-neglected metric for mobile devices). Having just one of these issues can be a game developer's worst nightmare as reviews will tend to focus on the one thing that we did poorly, ignoring all the things that we did well.“[51, S. 17]

14.3.5 Beispiel: Cyberpunkt 2077

"Cyberpunk 2077"dient als ein gutes Beispiel für die Konsequenzen von schlechter Performance. Vor dem Veröffentlichen war das Spiel eines der meisterwarteten Spiele von 2020. "CD Project RED", die Entwickler des Ti-

tels versprachen eines der größten Open-World-Action-Rollenspiele, das je auf den Markt kommen würde. Dabei wurde sehr viel wert auf die visuelle Qualität des Spieles gelegt, da das futuristische Aussehen ein wichtiger Teil des Marketings war. Die Stadt, in welcher sich die Spieler aufhalten, ist dabei äußerst detailliert und befüllt mit unterschiedlichen Aktivitäten, Autos, Charakteren, Waffen und weiteren Assets, welche dem Spiel sein einzigartiges Aussehen geben. Das Spiel verfügt über zahlreiche Grafikeinstellung als auch die Option Raytracing zu verwenden. Ein weiteres Ziel der Entwickler war es keine Art von Ladebildschirm während dem Spielen zu haben. Dies Bedeutet das Tausende von Assets konstant berechnet werden müssen. "Cyberpunk 2077ïst 70 Gigabyte groß und ist damit ein sehr rechenintensives Spiel. Ob das Spiel in einem guten Zustand laufen würde, ließ sich erst nach der Veröffentlichung klar sagen. Das Spiel sollte 2020 am 19. Dezember für Microsoft Windows, Playstation 4, Xbox One, GeForce Now und Google Stadia erscheinen. Hierbei waren die ersten Rezensionen geplagt von Performance Problemen auf allen erschienenen Geräten. Zahlreiche Windows Nutzer konnten trotz der richtigen Systemanforderung das Spiel nicht flüssig spielen.

Abbildung 14.9: Spiel Screenshot beim Launch

Es gab zahlreiche grafische Probleme und Fehler als auch konstante FPS-Einbrüche. Dies betraf vor allem Konsolenspieler auf der Xbox One und der Playstation 4. Das Spiel war nach ihrem Kauf für viele Käufer nicht spielbar, was zur sofortigen Rückgabe von über dreißigtausend Kopien führte. Weiters, entschied sich Sony Cyberpunk bis 21.Juni 2021 aus dem Sony Playstation Store zu entfernen.

„Cyberpunk 2077's disastrous launch led to very few refunds, according to CD Projekt Red CFO Piotr Nielubowicz, who shared details on the situation during an earnings call with investors. The dystopian cyberpunk RPG sold 13.7 million copies, but the developer only issued 30,000 refunds“

[57]

Cyberpunk 2077 beweist das Erlebnis der Spieler auf einem technischen Niveau stark von der Performance-Optimierung abhängt.

14.3.6 Fragestellung

Abgesehen vom tatsächlichen Gameplay und seinen Mechanics braucht ein Spiel Art und Assets, um seine eigene Identität zu definieren. Spiele sind deswegen mit visuellen Assets überflutet, um die Immersion der Spieler zu verstärken. Daraus folgt eine gigantische Anzahl an Mesh-Objekten, Texturen, Partikeln, Sprites und zahlreichen anderen Komponenten, die einen Großteil der Performance beanspruchen. Wie gelingt es Spieleentwicklern trotz des hohen Anspruches an grafischen Details, Realismus oder Umfang an Inhalt eine stabil laufende Experience zu kreieren?

14.3.7 Mesh Resolution

Anders als wie bei der Auflösung eines 2D Bildes geht es bei Mesh-Objekten nicht um die Anzahl der Pixel. Jedoch geht es in beiden Fällen um den sichtbaren Detailgrad. Im Falle von einem Mesh bezieht sich die Resolution um das Detail der Topologie. Wie bereits in Geometrie vs. Topologie klargestellt, handelt es sich bei der Topologie um die Anordnung der Faces. Die Auflösung bedeutet im Kontext eines 3D-Assets, wie viele Faces ein bestimmtes 3D-Mesh besitzt. Eine wichtige Grundregel für jedes Asset ist klarstellen, von welcher Wichtigkeit es ist. Assets, mit welchen die Spieler konstant interagieren oder im Verlaufe des Spieles häufig sehen, sollten den größten Detailgrad im Spiel besitzen, da hier ein Mangel an Qualität und Auflösung deutlich sichtbar ist. Die Auflösungen eines Assets spielt dabei eine große Rolle für die Performance in Spielen. Die Unity-Engine nimmt jegliche Assets so an, wie sie importiert werden, obgleich die Auflösung des Objektes notwendig ist. Das bedeutet, dass jedes existierende Face gerendert wird und damit sofortige Konsequenz für die Performance hat.[58, S. 16]

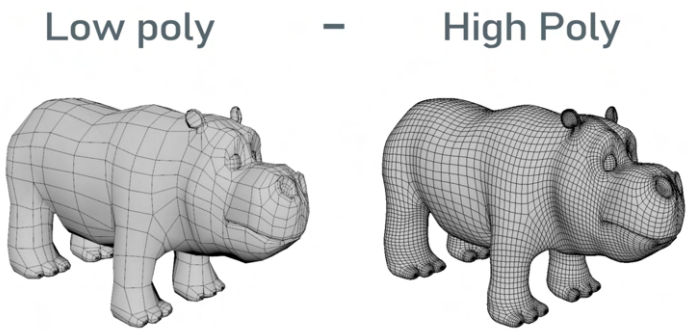


Abbildung 14.10: Mesh Resolution eines Assets

[59]

14.3.8 Mesh Cleanup

„Cleanen“ bedeutet für alle 3D-Assets, dass die momentane Auflösung kleiner wird. Für die Topologie heißt dies, dass die Anzahl an Faces reduziert wird.[51, S.152]Tripple-A-Spieleentwickler sind vor allem darauf angewiesen, alle ihre Assets zu Cleanen. 3D-Objekte, welche mittels Sculpting-Tools entstehen, sind so gut wie immer darauf angewiesen, ihre Topologie zu Cleanen.[58, S.14] Hier spricht man auch von der Retopologie. 3D-Softwares bieten unterschiedliche Methoden für die Retopolgie an. Programme wie Äutodesk Mayaöder "Blender"besitzen eine integrierte Funktion automatisch das Mesh zu „retopologizen“. Diese Funktion ist allgemein nützlich, die Auflösung zu reduzieren, ist jedoch nicht im Stande, einen geeigneten Edgeflow zu generieren.[58, S.15] Daher werden besonders wichtige Objekte meist per Hand bearbeitet.

14.3.9 Textur Auflösung und UV-Maps

Obwohl nicht konkret auf das 3D-Assets bezogen, ist UV als auch Textur Optimierung ein wichtiger Teil der weiterführenden Asset-Optimierung. Bei einer Textur handelt es sich um ein 2D-Bild, welches unterschiedliche Informationen beinhalten kann. Dabei ermöglicht es mehr Detail auf der Oberfläche eines Objektes darzustellen. Die Textur an sich kann durch unterschiedliche Methoden wie Fotografien, eingescannte Bilder, generierte Bilder oder auch prozedurale Vorgänge erstellt werden. Eine Textur hat dabei unterschiedliche Anwendungsmöglichkeiten und dient nicht immer schlicht für die Einfärbung eines Objektes. Darunter zählen Diffuse oder auch Albedo-Maps, Bump-Maps, Height-Maps, Normal-Maps, Metallic-Maps, Roughness-Maps und andere Arten.[58, S.18]

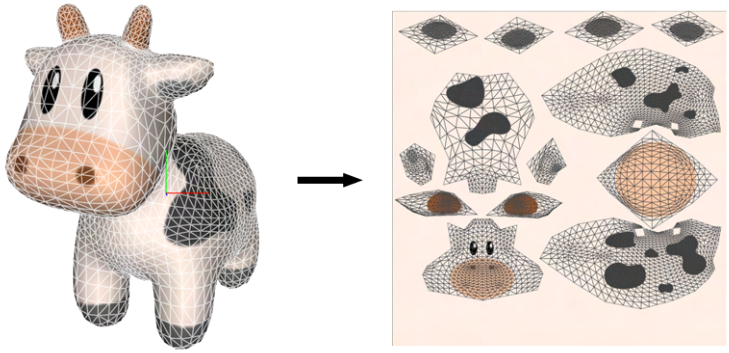


Abbildung 14.11: Textur eines Charakters

[60]

Texturen werden in 3D-Programmen und Game-Engines einem Material zugewiesen, welches als Aufgabe hat, die Texturen auf das 3D-Mesh zu projizieren. Da die Textur ein Bild ist, hat diese damit auch eine vom 3D-Artist definierte Pixel-Auflösung. Die Auflösung eines Bildes hat damit auch eine Speichergröße. Je höher die Auflösung, je größer die Speichergröße. Um die Speicherkapazität nicht zu überlasten und damit auch nicht Performance, sollte eine angemessene Textur Auflösung gewählt werden.[51, S.141] Kleine Objekte brauchen deswegen keine besonders große Textur. Eine UV-Map ist ein zweidimensionales Koordinatensystem, welches über die Oberfläche eines Meshes projiziert wird. U und V stellen die beiden Achsen dar und sind notwendig, wenn es darum geht, eine Textur korrekt auf einem Mesh zu übertragen.[61] 3D-Softwares ermöglichen den Nutzern mittels unterschiedlicher Tools das Mesh zu UV-Unwrappen. Dieser Prozess „faltet“ die Oberfläche aus und glättet sie, wodurch eine 2D-Fläche entsteht. Die UV-Map dient damit als Bauplan für alle Texturen des Models.[62, S. 79]

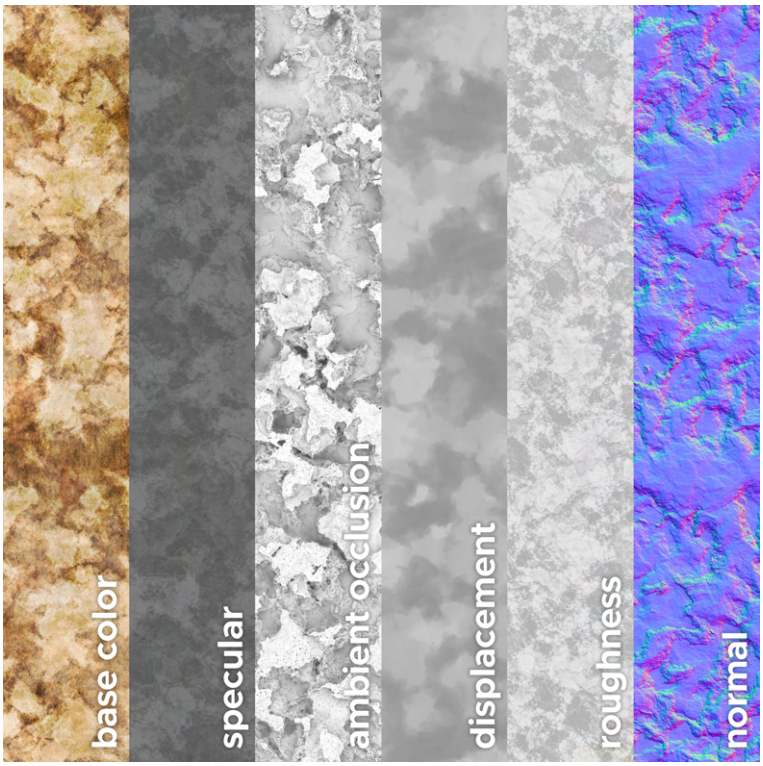


Abbildung 14.12: Unterschiedliche Arten von Texturen

[63]

Die UV-Map besteht dabei nicht aus einer einzigen großen Form, sondern wird meist auf UV-Islands aufgeteilt. Hierbei muss die Skalierung und Anordnung der UV-Islands beachtet werden, da der Platz, den eine Island einnimmt, direkt proportional zur Auflösung der Textur ist. Wichtige oder detaillierte Stellen des Meshes sollen deswegen mehr Platz in der UV als andere einnehmen, um Bildauflösung zu sparen.[58, S.18]

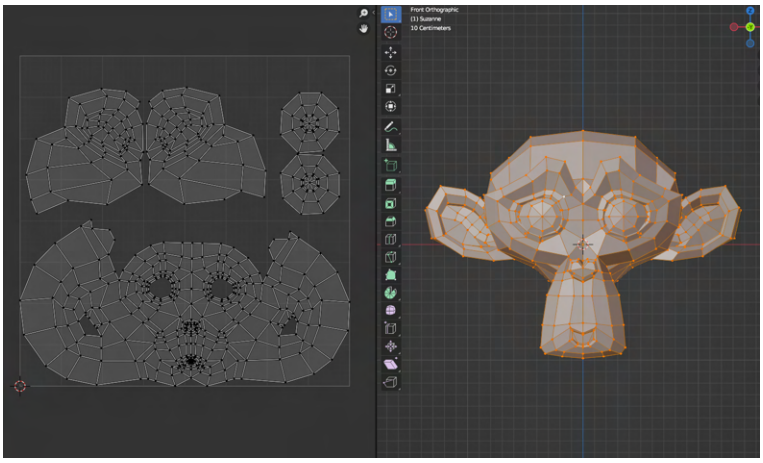


Abbildung 14.13: Beispiel für eine UV-Map

14.3.10 Texture Atlasing

„Atlasing is the technique of combining lots of smaller, isolated textures together into a single, large texture file in order to minimize the number of materials, and therefore draw calls, we need to use. This is effectively a means to exploit dynamic batching. Conceptually, this technique is very similar to the approaches of minimizing material usage that you learned in Chapter 3, The Benefits of Batching.“[51, S.145]

Texture-Atlases ist eine Technik, welche für reduzierte Draw-Calls und Unity-Batching oft eingesetzt wird. Da dieser Teil außerhalb des Arbeitsbereiches eines 3D-Artists ist, ist der technische Aspekt für die Umsetzung eines Texture-Atlases irrelevant. Einzig zu beachten ist, dass durch den Texture-Atlas weniger Draw-Calls hervorgerufen werden, wovon auch die Performance profitiert. Die Aufgabe eines Texture-Atlases ist es, eine Sammlung an kleineren Texturen auf eine einzige größere Textur zu bringen. Diese Technik ist nur dann anzuwenden, wenn alle Materials denselben Shader benutzen. In Essenz werden die einzelnen UV-Islands sämtlicher Objekte auf einer UV-Map organisiert und schließlich texturiert. Hiermit sparen sich die Entwickler einiges an Speicherplatz und können im Falle von kleineren Objekten eine größere Auflösung erhalten. Nachteil dieser Methode ist die spätere Bearbeitung der Textur.[51, S.146]

14.3.11 Billboarding

Ein Billboard (Plakatwand) ist nichts weiter als eine texturierte Fläche. In der Computergrafik ist die Billboarding-Technik eine Methode, um komplexe 3D-Assets zu vereinfachen. Als Beispiel betrachtet man einen Busch. Dieser

besteht aus vielen Wurzeln, Zweigen und Blättern. Da ein Busch jedoch in großen Mengen in Spielen auftaucht, ist es unpraktisch, ein wirkliches Mesh zu verwenden. Stattdessen wird die Projektion eines Busches oder einfach auch ein Sprite des Busches verwendetet und wird anstelle des Meshes eingesetzt. Damit wird eine immense Rechenleistung gespart, weil dieses Sprite aus nur einem einzigen Face besteht.[54, S. 48] Da ein Sprite zweidimensional ist, wird es aus bestimmten Blickwinkeln nicht sichtbar sein. Hier kommt das eigentliche Billboarding ins Spiel. Dabei wird zwischen zwei Methoden unterschieden. Die erste Methode benötigt ein einziges Billboard, welches sich immer in Richtung der Kamera rotiert, um seine Vorderseite anzuzeigen. Die zweite Methode verwendet mindestens zwei Sprites, welche sich überkreuzen. Dies lässt die einfache Fläche dreidimensional erscheinen. Für mehr Detail können mehrere überkreuzende Sprites verwendet werden. Da ein Busch jedoch nicht die gesamte rechteckige Fläche einer Plane überdeckt, werden Pixel nicht verwendet und wären damit besonders auffällig. Mithilfe einer Alpha-Mask lässt sich dies beheben. Eine Alpha-Maks ist eine besondere Form von Textur, welche mittels schwarz-weiß Werten die Transparenz der einzelnen Pixel definiert. Das PNG-Dateiformat eignet sich dabei besonders gut für Billboarding, da es einen eigenen Alphakanal besitzt, welcher nicht zusätzlich hinzugefügt werden muss.

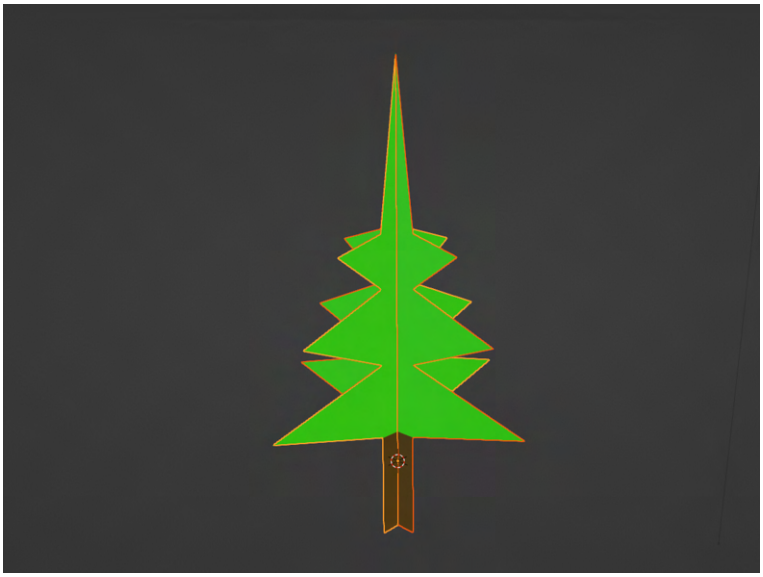


Abbildung 14.14: Einfaches Billboard

14.3.12 Shading

Das neue Mesh mit sauberer Topologie und geringer Anzahl an Polygonen ist nun für die Game-Engine nutzbar. Doch die geringere Auflösung hat

zur Folge, dass damit auch das Detail des gesamten Models reduziert wird. In vielen Fällen bedeutet dies, dass das Model besonders starke Kanten aufweist. Für runde oder weiche Objekte ist dies besonders unvorteilhaft. Spielen mit realistischer Grafik gelingt es dennoch, das Objekt richtig darzustellen, obwohl die Polygon-Anzahl deutlich verringert wurde. Dies liegt zum Teil am Smooth-Shading. Allgemein versteht man in der Computergrafik unter Shading die Oberflächeneigenschaften eines 3D-Meshes. Es bezieht sich dabei auf die geworfenen Schatten des Objekts und dessen Reflektions - und Lichtbrechungseigenschaften. Dabei soll das Shading bestimmte visuelle Effekte generieren, welche beim wirklichen Licht auftreten. Dazu werden tatsächliche Lichteigenschaften mit komplexen Algorithmen erschaffen. Durch diese Algorithmen lassen sich in Form einer digitalen „Illusion“ der Interpolation die Oberfläche von Objekten glatter darstellen. Hieraus wurden deshalb unterschiedliche Techniken entwickelt, die ein 3D-Model unterschiedlich darstellen. Die weitaus simpelsten Techniken sind dabei Flat Shading und Smooth Shading.

14.3.13 Flat Shading

Flat-Shading ist die einfachste Methode, einen Gegenstand darzustellen und ist in 3D-Softwares der eingestellte Standard. Dafür wird jedem Face eines Mesh eine einzige Farbe zugewiesen. Diese Farbe wird in Abhängigkeit vom Licht durch den Winkel und dessen Flächennormale (Normalvektor des Faces) bestimmt. Dies führt zu einer zweidimensionalen und daher auch flachen Erscheinung, bei welcher die Kanten zu scharfen Übergängen zwischen den Faces führen. Deswegen erscheint das Objekt bei einer niedrigen Auflösung nicht besonders realistisch. Dafür könnte man mit einer höheren Auflösung kompensieren, wenn die Rechenleistung nicht berücksichtigt wird. Der Vorteil des Flat Shading ist, dass diese Technik besonders einfach und schnell zu rendern ist. Somit ist sie besonders performant.

14.3.14 Smooth Shading

Im Kontrast zum Flat Shading versucht Smooth Shading die Oberfläche eines Objektes glatter darzustellen. Dafür ist die Interpolation notwendig, um die Flächen der Faces zu glätten. Hierzu gibt es zwei übliche Methoden, um dieses Ziel zu erreichen.

Gouraud Shading

Henri Gouraud stellte diese Technik erstmalig 1971 vor. Diese Methode konzentrierte sich auf die Vertices eines Polygons und interpoliert basierend auf dem Licht die Oberfläche. Die wichtigsten Parameter für diesen Effekt sind die Vertices, deren Normalvektoren, die Vektoren des Lichtes und auch die

Kamerarichtungsvektoren. Damit wird schlussendlich die Farbe der einzelnen Vertices berechnet, welche nur noch interpoliert werden müssen. Die Interpolation ist eine mathematische Technik, mittels welcher die Werte zwischen zwei bekannten Werten berechnet werden. Damit ist es nur eine Annäherung an die gegebenen Werte. Die bisherige flache Oberfläche des Objektes ist nun glatt, wobei der Umriss des Objekts weiterhin kantig ist. Gourand Shading braucht dabei eine größere Rechenleistung als Flat Shading, ist jedoch um einiges performanter als einer Erhöhung der Auflösung. Deswegen ist sie standardmäßige Shading-Methode für glatte Oberflächen in Realtime.

Phong Shading

4 Jahre nach dem Gouraud Shading wurde die Phong-Shading Technik von Bui Tuong Phong vorgestellt. Ähnlich wie bei der bisherigen Methode liegt die Konzentration erneut bei den Vertices des Objektes. Diesmal werden die Normalen der Vertices berechnet und anhand dieser wird bei der Färbung der zwischen den Normalen der Eckpunkte eine neue interpolierte Normale erschaffen. Die Lichtquelle, Blickrichtung und der Glanz des Materials sind dabei wichtige Komponenten. Das resultierende Smooth Shading führt zu einem realistischeren und akkurateren Ergebnis für die geglättete Oberfläche. Da diese Berechnung jedoch um einiges komplexer ist, ist sie deswegen um einiges rechenintensiver als das Gourand Shading. Daher wird sie selten bei Realtime Rendering benutzt.

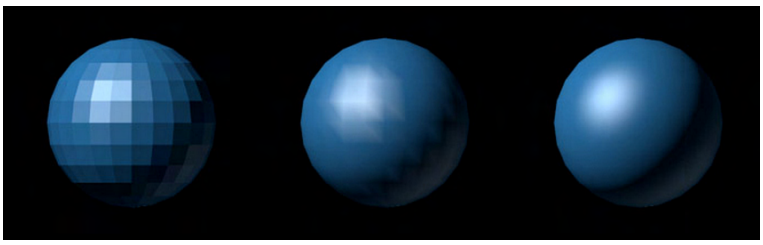


Abbildung 14.15: Verschiedene Arten von Shading: Flat - Gouraud - Phong

[64]

14.3.15 Detail, ohne auf Performance zu vergessen

Betrachtet man die Welt um sich herum, stellt man fest, dass jedes Objekt mikroskopisch kleine Details hat. Spieleentwickler und 3D-Artist können sich technologisch nicht leisten, jedem Objekt seine atomaren Feinheiten zu geben, jedoch sind essenziellen Merkmale wichtig, um das Objekt als dieses zu erkennen. Betrachtet man z. B. das Gesicht eines alten Mannes, so erkennt

man viele Falten, Narben oder raue Hautflecken. Mit dem bisherigen Wissen müsste man all diese Feinheiten modellieren oder sculpten, um den alten Mann als diesen zu erkennen.[56, S. 3] Die bisherigen Prinzipien erwarten jedoch, dass das Mesh eine möglichst geringste Polygonanzahl besitzt. Die Falten würden dagegen verstoßen, da sie um einiges mehr Polygone benötigen, um richtig auszusehen. Glücklicherweise können die heutigen Materials in 3D-Softwares und Game-Engines mehr als nur Farbwerte nutzen. Falten und ähnliche Feinheiten geben einem Mesh Tiefe, welche das Objekt dreidimensionaler aussehen lässt. Dafür verwendet man entweder Bump -oder Normal-Maps.

> „The bump, normal, and parallax occlusion mapping shaders add 3D depth to an otherwise flat surface. Bump maps are grayscale and display the most limited 3D effect; the others add depth using a color map with lighting information encoded in it. These shaders are all similar in what they accomplish, but depending on the exact code of the shader and the supporting hardware, the effect can range from really cool to absolutely awesome. A basic normal map adds a level of depth deeper than the bump map, but more advanced forms of these shaders add details and behaviors such as self-occlusion, silhouetting, and self-shadowing. Figure 3.23 is a simple visual demonstration of how the shader operates. Because we can calculate the light of the surface for every pixel, we don't need to include geometry to create shadows and highlights as we did when we lit per polygon. We can now tell the 3D application to treat each and every pixel as if it were reacting to light, as it did when it was on the high-polygon model.“ [65, S. 96]

Bump-Maps

Der Ausdruck Bump-Map lässt sich in der Realität betrachten, indem man einen Finger über die Oberfläche eines Gegenstandes fährt. Man wird kleine Unebenheiten auf jeder Oberfläche spüren und auch sehen. Für die Computer-Grafik ist die Bump-Map notwendig, um diese Unebenheiten zu visualisieren. Dabei handelt es sich nur um eine 2D-Textur aus Grauwerten variierend zwischen 0 (schwarz) und 225 (weiß). Diese Werte stellen die Höheninformation der Textur dar. Mittels komplexer Berechnungen können damit Höhenunterschiede generiert werden, welche aus unterschiedlichen Blickwinkeln betrachtet werden können. Bump-Maps ist mittlerweile eine veraltete Methode, Detail zu generieren, da sie wegen ihrem kleinen Farbspektrum wenig Informationen beinhalten kann.[62, S. 86]

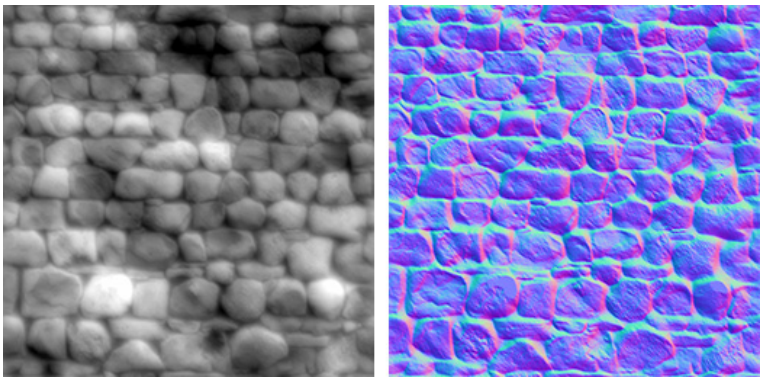


Abbildung 14.16: Unterschied zwischen Bump-Maps und Normal-Maps

[66]

Normal-Maps

Mittlerweile ist eine bessere Methode hervorgekommen: Normal Mapping. Genauso wie Bump-Mapping zielt Normal Mapping darauf ab, mehr Detail in der Schattierung ohne enorme Rendering-Kosten herauszuholen. Diese orientiert sich nicht an Grauwerten, sondern verwendet das RGB-Farbspektrum. Der Vorteil davon ist, dass mehr Lichtinformationen aus den Farben entnommen werden können, weswegen bessere und realistischere Ergebnisse produziert werden können. Diese RGB-Werte entsprechen jeweils den dementsprechenden X, Y und Z Koordinaten. In Essenz ermöglichen Normal Maps bessere Ergebnisse jedoch lassen sie sich per Hand nicht ansatzweise so einfach zeichnen wie Bump-Maps. Anstatt sie per Hand zu zeichnen, erlaubt der Fortschritt der Technologie einen automatisierten Generierungsprozess. Adobe Photoshop kann mittlerweile aus ausgewählten Bildern eine Normal Map generieren. Jedoch kommt es zu ungenauen bzw. fehlerhaften Ergebnissen. Diese Methode ist vor allem unvorteilhaft, wenn wir einen weniger generischen Gegenstand bspw. eine Holzwand betrachten. Dafür werdet man einen weitaus besseren Vorgang: das Texture-Baking.

14.3.16 Texture Baking

Texture-Baking ist darauf ausgerichtet, den Detailgrad eines Highpoly-Meshes auf ein Lowpoly-Mesh zu übertragen. Da viele Assets nicht auf generischen Texturen basieren, ist dies nützlich. Auch dann, wenn man ein bestimmtes Aussehen erreichen möchte.[56, S. 8] Wie bereits erwähnt, braucht man dafür eine hoch aufgelöste und niedrig aufgelöste Version des Meshes. Unterschiedliche Programme wie SSubstance Painter", "Mudboxöder "Blenderërmöglichen diesen Prozess einfach zu gestalten. Dafür sind lediglich beide Models auf derselben Position notwendig, die UV-Maps beider Objekte und danach

folgt die automatisierte Berechnung des Computers. Dieselben Positionen in einem Koordinatensystem sind deswegen notwendig, weil die genaue Lichtberechnung im Verhältnis der Postionen konvertiert, werden müssen. Das Highpoly Mesh schießt dabei Raycasts aus, welche dann die Oberfläche des Lowpoly-Meshes treffen. Anhand dieser Treffpunkte wird nun eine Normal-Map generiert, welche Lichtinformationen des Highpoly-Objekt basierend auf der UV-Map des Lowpoly-Mesh baked.

Kapitel 15

Theoretische Implementation in Tricks ´n´ Treats

Als ein asymmetrisches Couch-Party VR Spiel gibt es bei "Tricks ´n´ Treatsëinige Faktoren im Bezug zu Performance zu beachten. Das Spiel hat sowohl die Kamera der PC-Spieler als auch die zwei notwendigen Kameras für den VR-Spieler zu rendern, womit jegliche Assets in Realtime dreimal gerendert werden müssen. Viele der Performance-Techniken, die erwähnt wurden, können dabei in gemeinsam oder als Ersatz füreinander auftauchen.

15.1 Bedenken bei der Implementation

15.1.1 Einschränkung durch den Artstyle

"Tricks ´n´ Treats"grafisches Aussehen ähnelt nicht der Realität. Es ist von den modernen und stilisierten Grafiken aus Spielen wie aus SSuper Mario Odysseyünd "The Legend of Zelda Breath of the Wild"(BotW) inspiriert. SSuper Mario OdysseysÄssets besitzen übertrieben Proportionen und weichen Kanten. "BotW"nutzt bekanntlich Cell-Shading, um mit nur einfachen Farben und Texturen ein zeitloses Aussehen hervorzubringen. "Tricks ´n´ Treats"versucht beide diese Eigenschaften zu verbinden und in Lowpoly-Form umzusetzen.

15.1.2 Unbedingt Notwendig

Mesh Cleanup

Für "Tricks ´n´ Treats"wird es unausweichlich sein, die Topologie der Assets zu säubern. Wie bereits erwähnt, bedeutet dies, dass die Mesh-Auflösung reduziert wird. Grund dafür ist, dass es die einfachste Methode ist, um die Performance zu verbessern. Grundlegend würde man dafür jedes Model vor dem UV-Unwrappen und Texturieren erneut auf unnötige Faces oder Edge-

Loops überprüfen. Für einfache Objekte, wie bspw. einen Eimer eignet es sich besonders, sofort mit so wenig Geometrie wie möglich anzufangen und nur wenn notwendig mehr Detail hinzufügen. Im Falle der spielbaren Charaktere der PC-Spieler würde man jedoch umgekehrt vorgehen. Die Charaktere besitzen weiche Kanten, weswegen es einfacher ist, die Auflösung im Nachhinein hinunter anstatt hinaufzuskalieren. Der offensichtliche Vorteil des Mesh-Cleanups ist, dass im Spiel weniger Faces gerendert werden müssen. Deswegen wird dieser Prozess unvermeidlich sein, da es technisch keine Probleme bereitet und einfach implementierbar ist.

Smooth Shading

Jedoch erhalten wir damit ein visuelles Problem, da die Assets nun kantig sind. Da "Tricks ´n´ TreatsÄssets stilistisch weiche Kanten haben soll, kann das Mesh-Cleaning nicht allein eingesetzt werden. Deswegen sollte man zusätzlich jedes 3D-Asset „Smooth-Shaden“. Damit sind weiche Kanten und Formen möglich, ohne eine enorme Rendering-Last zu werden. Diese beiden Methoden werden damit gemeinsam die Essenz der Asset-Optimierung für "Tricks ´n´ Treats"bilden, da sie keine wirklichen Nachteile besitzen.

15.1.3 Hat die Methode einen Sinn?

LODs und Billboarding

Aufbauend auf der Mesh-Auflösung würde sich das Unity-LOD-System (Level of Detail) eignen. Das System verwendet dafür unterschiedliche Models mit unterschiedlichem Detailgrad, welche in Abhängigkeit von der Entfernung der Kamera ausgetauscht werden. Befindet sich ein Spieler nahe an dem Objekt, so wird der LOD das hochaufgelöste Model gerendert. Je weiter weg sich der Spieler befindet, desto weniger Auflösung ist nötig, um denselben Gegenstand darzustellen. Für die 3D-Assets würde dies bedeuten, dass mehrere Versionen vom selben Objekt erstellt werden müssen.

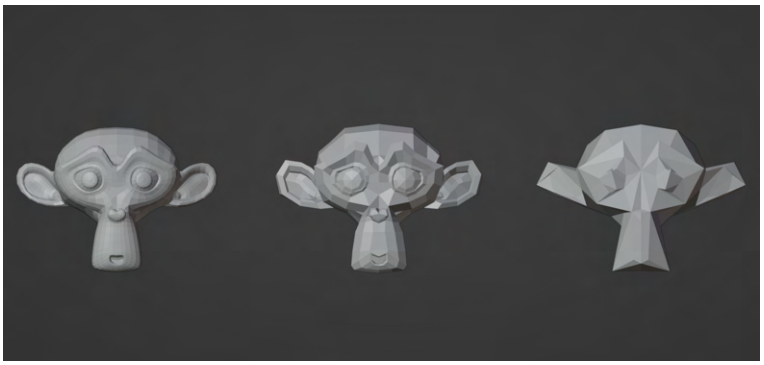


Abbildung 15.1: Verschiedene Auflösungen des selben Meshes

Das Prinzip des Billboarding würde sich kombiniert mit LODs zusätzlich für die Performance eignen. Dies würde bedeuten, dass anstelle eines Meshes zwei oder mehrere überlappende Bilder verwendet werden. Anstelle eines niedrigdetaillierten Models würde man als niedrigste LOD-Stufe das Bild verwenden und könnte damit immense Rendering-Kosten sparen. Das liegt daran, dass nur noch ein oder zwei Faces anstatt einem gesamten Model gerendert werden müssen. Diese Methode könnte an sich für bessere Performance sorgen und die Grafikkarte entlasten. Die Grafikkarte kann jedoch nur deswegen entlastet werden, weil die CPU dafür sorgt, dass die LOD-Stufen ausgetauscht werden. Dies führt jedoch zu einer größeren Rechenlast für die CPU. Da "Tricks ´n´ Treats"drei Kameras hat, würde sich die Rechenlast enorm vergrößern und bestehende Gleichgewicht zwischen CPU und GPU würde entarten. Die GPU wäre, indem fall unterfordert, die CPU hingegen überfordert, wodurch die CPU alle aufgestauten Berechnungen zuerst beenden müsste. Damit gebe es um einiges mehr Nachteile als Vorteile im Falle von "Tricks ´n´ Treats".

Billboarding in Tricks ´n´ Treats

Billboarding allein würde in "Tricks ´n´ Treats"keinen wirklichen Vorteil bieten, da diese Technik hauptsächlich für Foliage in Spielen genutzt wird. "Tricks ´n´ Treats"besitzt keine wirklichen Pflanzen oder Büsche. Möglicherweise würde man Billboarding für gewisse dekorative Assets auf der Strecke verwenden, wenn es sich nur um ein PC-Spiel handeln würde. Da der VR-Spieler jedoch die Assets auf der Strecke von überall sehen kann, würde der visuelle Effekt von Billboarding nicht zustande kommen. Billboarding würde damit immer noch performant sein, jedoch nicht das Ziel erfüllen, für welches es optimiert wurde.

15.1.4 Texturing in Tricks ´n´ Treats

Handgemalte Texturen vs. einfache Farben

Die Assets in "Tricks ´n´ Treats"werden größtenteils Texture-Atlases verwenden, um Speicherplatz als auch Draw-Calls zu sparen. Dabei werden diese Texturen meist aus einfachen Farben bestehen oder per Hand gezeichnet. Assets, die nicht im Hintergrund stehen oder besonders klein sind, brauchen keine gezeichnete Textur, da es nicht auffällt, wenn sie simpel texturiert werden, wodurch zusätzlich an Texturauflösung gespart werden kann. Daher würden die spielbaren Charaktere oder die Assets für die Strecke nur einfach texturiert werden. Dabei liegt ein großer Fokus darauf, dass die UV-Map aller Objekte so performant wie möglich genutzt wird, damit man die Auflösung der Textur noch weiter reduzieren kann. Die Assets im VR-Raum, wie bspw. Items und der Arbeitsplatz des VR-Spielers können hingegen handgezeichnete Texturen verwenden. Dekorative Objekte können dennoch auf

einer einzigen Textur texturiert werden. Hierbei gilt es besonders auf die Qualität der UV-Map zu achten.

Atlasing in Tricks ´n´ Treats

Das Problem an Texture-Atlases ist, dass diese Methode um einiges Zeitintensiver ist als für jedes Objekt eine einzelne Textur zu erstellen. Das liegt vor allem daran, dass trotz der gruppieren aller UV-Maps eine gute Organisation für die Atlases gebraucht wird. Bspw. sollten die Item-Texturen von den Charakter-Texturen getrennt ein. Auch mit guter Organisation wird um einiges mehr Zeit gebraucht, um die Texturen zu bearbeiten. Der notwendige Zeitaufwand sollte es doch im Falle von "Tricks ´n´ Treats"wert sein, da man hiermit an Anzahl an Texturen, Rendering-Kosten, Speicherplatz und Auflösung sparen kann.

15.1.5 Texture Baking als Alternative

Eine Alternative zur LOD-Methode ist das Texture-Baking. Diese könnte zusätzlich zu den Texturen eingesetzt werden, um die Faceanzahl zu reduzieren. Einkerbung in Holz oder die Rinnen einer Ziegelsteinwand könnten mithilfe dieser Methode simpel dargestellt werden. Ähnlich wie bei LODs werden dafür Varianten von demselben Asset benötigt. Im Falle vom Texture-Baking, braucht man eine Highpoly -und eine Lowpoly-Version vom selben Mesh. Die Details des Highpoly-Meshes werden schließlich auf einer Normal-Map generiert und übertragen. Nun kann das Highpoly-Mesh verworfen werden und gegen das Lowpoly-Mesh ausgetauscht werden, welches mit der angehängten Normal-Map beinahe identisch aussehen sollte.

Normal Map oder Texture Atlas

Obwohl Normal-Maps die Anzahl an benötigten Faces reduzieren, sind sie nicht immer vorteilhaft. Wird ein oder mehrere Texture-Atlase benutzt, so muss man für jedes Objekt eine Highpoly -und Lowpoly-Verion erstellen, diese UV-Unwrappen und schließlich Texture-Baken. Dies ist ein besonders langwieriger Prozess, welcher manuell ausgeführt werden muss. Bei einzelnen Objekten mit eigener Textur würde sich Texture-Baking besser eignen als das Atlasing. Somit sind Normal-Maps in "Tricks ´n´ Treats"meist unpraktisch, außer man würde die Atlases durch einzelne Texturen austauschen.

Kapitel 16

Praktische Implementation in Tricks ´n´ Treats

16.1 Endresultat

16.1.1 Entwicklung des Arbeitsprozesses

Das tatsächliche Implementieren verlief nicht auf einem linearen Weg. Oftmals wurden die Prioritäten der Assets neu bewertet und hinsichtlich auch die Performance-Optimierung. Am Anfang des Projektes waren viele der Methoden nur ein grober Gedanke, da viele der visuellen Ideen und Entscheidung erst mit der weiterführenden Entwicklung des Projektes entschieden worden sind. Daraus erschloss sich, dass manche Methoden zur Performance-Optimierung vollkommen unpraktisch waren oder noch genauer durchgeführt wurden.

16.2 Erwartete Ergebnisse

16.2.1 Mesh Cleaning in Tricks ´n´ Treats

Die Verbesserung und Optimierung der Mesh-Topologie war wie erwartet am einfachsten zu implementieren und einzusetzen. Neben dem tatsächlichen Modellieren beruhte ein überschaubar großer Teil des Arbeitsaufwandes, alle Assets zu optimieren. Dabei wurde festgestellt, dass dieser Prozess auf zwei Arten umgesetzt wurde. Vor der Implementation war die Annahme, dass unwichtige Objekte mit wenig Geometrie anfangen und bei Bedarf mehr Detail erhalten würden. Assets wie bspw. die Charaktere hingegen fingen mit einem detaillierten Mesh an und wurden dann optimiert. Obwohl dieses Prinzip angewendet wurde, gab es beim Modellieren ein weiteres Kriterium zu beachten. Abhängig davon, ob es sich um ein organisches Objekt handelte oder nicht, wurde die „Downscaling“ Methode verwendet. Der Grund dafür ist dass Assets mit weichen Kanten auf diese Art schneller modelliert wurden

und auch besser aussahen.

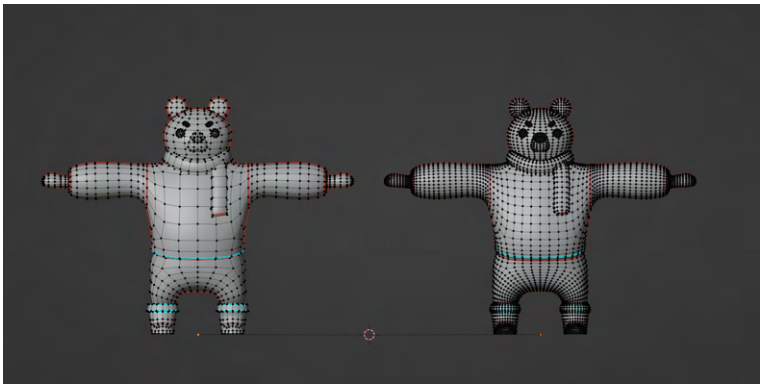


Abbildung 16.1: Vergleich: 1200 Faces vs 6200 Faces

Unsichtbare Faces und Clipping

Um die Performance der Models noch weiter zu verbessern, wurde zusätzlich Faces gelöscht, welche sich schneiden (Clipping) oder für die Spieler nicht sichtbar waren. Diese Methode war beispielsweise für die Charaktere und dekorative Assets in besonders vorteilhaft, da man überschüssige Faces somit entfernen kann. Betrachtet man den Hasen-Charakter, so stellt man fest, dass der Kopf, die Ohren und der Torso nicht aus einem einzigen Objekt bestehen. Diese drei Körperteile wurden einzeln erstellt und zusammengesteckt. Viele Faces würden deswegen nie für den Spieler sichtbar sein, weswegen man sie entfernen kann. Da alle Charaktere auf Snowboards stehen, brauchen sie auch keine Faces an ihren Fußsohlen.

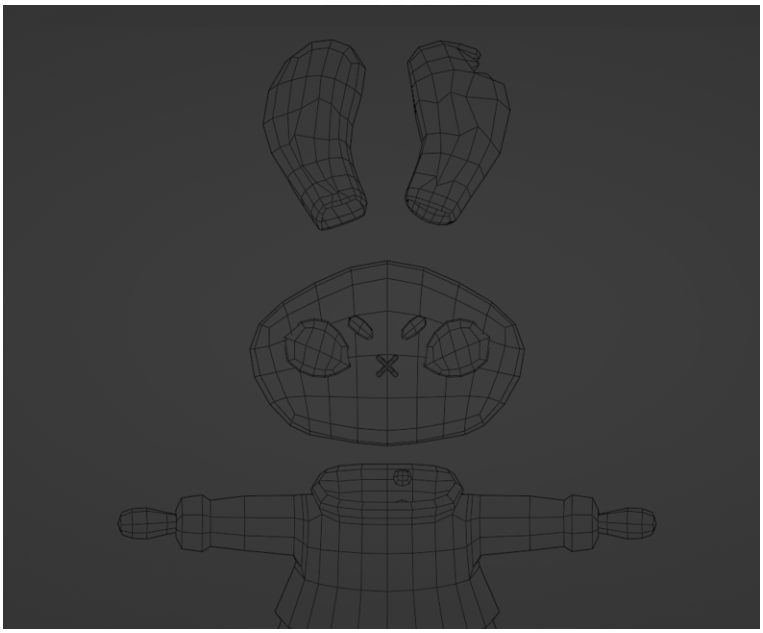


Abbildung 16.2: Unötige Faces

Ebenso lassen sich von vielen Objekten die Boden-Faces löschen, da diese auch nicht sichtbar sind. Töpfe, Tischbeine oder auch Bücher haben deswegen keine Faces am Boden und Rücken. Bei einem einzelnen Objekt würde sich dieser Arbeitsaufwand wahrscheinlich nicht auszahlen, weil meistens nur sehr wenige Faces gelöscht werden. "Tricks n' Treats"besteht jedoch aus einer großen Szene mit vielen Objekten, weshalb es sich in diesem Fall auszahlt, alle unsichtbaren Faces zu löschen, obwohl es etwas Zeitaufwendiger ist. Auf diese Art wurden ca. 1.000 Faces gelöscht, wodurch man ungefähr 3.000 Faces weniger rendern musste.

16.2.2 Smooth Shading in Tricks ´n´ Treats

Da die Assets in "Tricks n'Treatsëine niedrige Mesh-Auflösung besitzen, hatten alle Objekte ein sehr kantiges Aussehen. Um dies zu verhindern, wurde wie erwartet Smooth-Shading verwendet. Obwohl das Projekt eine große Anzahl an Assets hatte, war diese Methode dennoch um einiges weniger rechenintensiv als allen Assets ein detaillierteres Mesh zu geben.

Abbildung 16.3: Vergleich: Flat Shading vs. Smooth Shading

Auto Smooth – Das Beste aus beiden Welten

Das Ziel dieser Shading-Einstellung ist es, für eine glatte Kanten zu sorgen. Dabei entsteht jedoch das Problem, dass ein Objekt sowohl weiche als auch scharfe Kanten haben kann. Glücklicherweise ermöglicht die 3D-Software Blender, sowohl weiche als auch scharfe Kanten zu haben. Die Auto-Smooth Funktion ermöglicht es, manuell einzustellen, ab welcher Neigung die Edges weich sein sollen. Zusätzlich ermöglicht Blender, einzelne Edges auszuwählen und scharf zu rendern. Diese Einstellungen gehen beim Importieren in Unity nicht verloren und benötigt keinen großen Zeitaufwand

16.3 Texturieren – Die Evolution

Vor allem beim Texturieren gab es einige Änderungen, welche von der eigentlichen Idee abschweifen. Viele der geplanten Ideen wurden wegen Zeitmangel oder auch Performancegründen komplett verworfen. Eine wichtige Design-Entscheidung, welche ungefähr in der Mitte der Projektentwicklung getroffen wurde, war es nur noch einfache Farben für alle 3D-Assets zu verwenden. Da die Erstellung einer einzelnen Textur viel Zeit in Anspruch nahm, wurde sicherheitshalber beschlossen, nur Farben zu verwenden. Damit wurde zusätzlich auch indirekt Performance verbessert, da es nun um einiges weniger Texturen gab, womit weniger Speicherplatz genutzt wurde und es weniger zu rendern gab. Zusätzlich zu dem Ganzen wurden sämtlich Texturen mit einem Cell-Shader gerendert. Diese Änderung war ebenfalls der Grund, warum kein Texture-Baking oder Normal-Maps verwendet wurden, da die Normal-Maps

das simple Aussehen des Cell-Shaders nicht komplementierten.

16.3.1 Textur Paletten

Texture-Atlasing ist die hauptsächliche Methode zum Texturieren in Tricks n'Treats. Insgesamt werden nur drei Texturen verwendet, welche alle für eine jeweils spezifische Gruppe an Assets verwendet werden. Man unterscheidet zwischen: Raum-Textur: Diese Textur ist für alle Assets im VR-Raum zuständig Charakter-Textur: Diese Textur ist für die spielbaren Charaktere zuständig Asset-Textur: Diese Textur ist für alle Assets auf der Strecke zuständig. Diese Texturen sind jedoch eigentlich nur eine simple Farbpalette und bestehen nur aus wenigen Pixeln. Da alle Assets nur eingefärbte Faces haben, muss man nicht großartig auf die UV-Map und Textur achten. Da eine Farbe eigentlich nur ein einziger Farbwert ist, kann dieser in einem einzigen Pixel dargestellt werden. Die UV-Map aller Objekte kann deswegen automatisch unwrapped werden, wobei sich die einzelnen Faces der UV-Map dann in einem einzelnen Farb-Pixel der Palette befinden. Die Raum-Textur ist dabei die größte Textur mit insgesamt 30x30 Pixeln. Mit dieser Methode fällt das UV-Mapping als auch Texturieren so gut wie weg und bietet dennoch die Vorteile eines Texture-Atlases mit nur sehr wenig benötigtem Speicherplatz. Das einzige Problem, welches bei Technik vorkommt, ist eine etwas langwierige Veränderung. Es ist oftmals vorgekommen, dass mehrere Objekte denselben Farb-Pixel teilten. Wenn eine bestimmte Farbe für ein Asset nicht angemessen war und die Farbe verändert wurde, so wurde die Farbe für mehrere Assets gleichzeitig verändert, weswegen man nun allen betroffenen Assets neue Farben zuweisen musste.

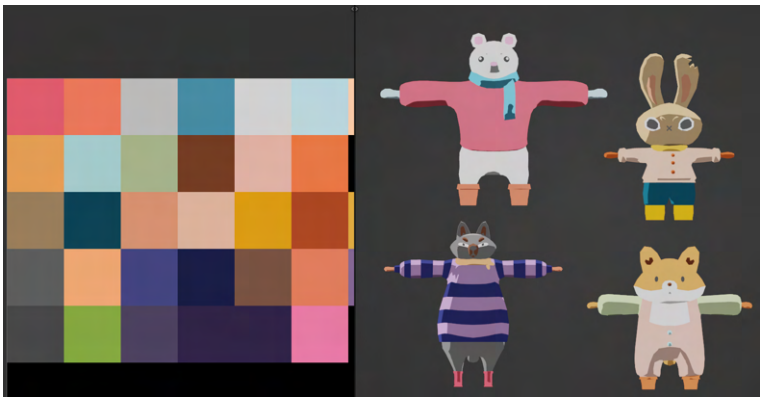


Abbildung 16.4: Textur Palette und Charaktere

Eine gute Erweiterung dieser Methode wäre es wahrscheinlich, eine vorzeitliche bessere Organisation und auch Kategorisierung der Assets. Damit hätte man bspw. die Asset-Palette und Raum-Palette auf nur eine Textur

komprimieren können. Zusätzlich könnte darauf geachtet werden können, dass eine Farbe nicht mehrmals in derselben Textur vorkommt oder dass eine Reihe an Pixeln nur für ein Objekt zuständig ist. Nachdem alle Assets texturiert sind, hätte man außerdem noch die Textur auf die genaue Anzahl an verwendeten Pixeln reduzieren können.

16.4 Rückblick

Mit dem Ende des Projektes lassen sich im Nachhinein manche Optimierungsmethoden erneut betrachten. Das allgemein größte Problem während der Projekt-Ausführung ist ein Mangel an Organisation und eine Überschätzung des benötigten Zeitaufwandes. LODs und Billboards als Methoden, um für eine bessere Performance zu sorgen, sind jedoch nicht darauf ausgelegt, als eine Kombination aus PC und VR-Spiel zu funktionieren. Es wäre weitaus wichtiger gewesen, das Mergen von Objekten mehr zu beachten. Da jedes Assets gerendert werden muss, hätte man weniger Rendering-Aufrufe gebraucht, wenn viele kleinere Objekte zu einem einzigen Mesh zusammengetan wären.[51, S.154]

Kapitel 17

Conclusio

Die Performance-Optimierung von 3D-Assets ist ein wichtiger Teil des Game-Developments und auch 3D-Modelings. Die Beherrschung dieser Technik erleichtert einen großen Teil des Entwicklungsprozesses und ermöglicht ein besseres Spielerlebnis. Vor allem in der Entwicklung von VR-Spielen wie in diesem Projekt ist dies sehr wichtig. Selbst in der einfachsten Form der Optimierung lässt sich "Tricks n'Treats"hiermit verbessern. Da sich die Optimierung von Spiel zu Spiel nur geringfügig unterscheidet, ist der Optimierungsprozess besonders einfach zu implementieren. Im Falle, dass "Tricks n'Treatsïn Zukunft weiterentwickelt wird, sollte hierbei ein mehr auf die Vor- und Nachteile der verschiedenen Methoden geachtet werden. Doch im Allgemeinen lässt sich ein zufriedenstellendes Ergebnis bei der bisherigen Implementation sehen.

Anhang A

Projektdokumentation

A.1 Meilensteine

A.1.1 Prototype (23.09)

Die Hauptmechaniken wurden Implementiert, und erste Systeme Getestet. Der Art-Style von unserem Spiel wurde ausgearbeitet, und dazu wurden erste Style-Guides, Charaktere, Farbpaletten und Target-Experience festgelegt.

A.1.2 Iteration Interim 1 (20.10)

Es wurden die Spell-Types und des Raum-Design definiert und implementiert. Der Gameloop des Spieles funktionierte erstmals komplett. Es wurden außerdem erste Playtests in einem Test-Level durchgeführt, um mögliche Probleme mit dem Design herauszufinden.

A.1.3 Iteration Interim 2 (22.12)

Ein Großteil der Grafiken und 3D-Modelle wurden importiert. Außerdem wurde die Mana-Menge und andere Variablen anhand von Playtests, die durchgeführt wurden, um früher auf Fehler im Design zu stoßen und sie zu beheben, angepasst. Die Technik des Spieles ist auch größtenteils fertiggestellt.

A.1.4 Vertical Slice (30.03)

Ein Vertical Slice des Projekts wurde finalisiert. Der Gameloop ist nun ohne Errors durchspielbar, und die gewünchte Experience, Art/Design und Performance wurden gut umgesetzt. Es gibt außerdem ein vollständiges Presskit inklusive Trailer.

A.2 Playtest Protokolle & Ergebnisse

Es wurden bereits mehrere Playtests durchgeführt. Die Ersten mussten aufgrund unerwarteter Fehler frühzeitig abgebrochen werden. Die Playtests, die Problemlos abgelaufen sind, haben allerdings die gewünschte Experience erzeugt. Nach den Playtests, wurden die Teilnehmenden anschließend darum gebeten einige Fragen zu beantworten.

A.2.1 Welcher Spell ist der Lustigste?

Der Großteil der PC- als auch der VR-Spielenden gaben an, dass die Bombe der beste Spell sei. Auf Platz zwei war die Eis-Fläche, da damit gute Kombinationen erzielt werden konnten.

A.2.2 Welcher Spell ist der Unlustigste?

Die Hagel-Wolke wurde von vielen Spielenden als „sinnlos“ oder „Unfair“ beschrieben. Das ist wahrscheinlich darauf zurückzuführen, dass sie erst relativ spät Einfluss auf die Snowboarder hat und nicht sehr genau platziert werden kann, also nicht wirklich ein Zusammenhang von Wolken-Kill zu VR-Skill besteht.

A.2.3 In welcher Rolle hattest du mehr Spaß?

Fast alle Spielende, die beide Rollen spielen konnten, gaben an, dass sie als als Magier mehr Spaß hatten.

A.2.4 Gab es Momente wo du stark mitgefiebert hast?

Die Spannung wenn nur nur noch ein Snowboarder im Spiel ist, ist sehr intensiv. Vor allem gegen Ende einer Runde.

A.2.5 Wie würdest du die Gesamterfahrung bewerten?

Der Durchschnittliche Score der Gesamterfahrung beträgt 8/10 Punkten. „Ist für ein paar runden ganz witzig.“

A.2.6 Gab es Momente in denen dir langweilig war?

Die VR-Spielenden gaben das Aufsetzen von SteamVR und das Warten auf die Snowboarder während sie ihre Charaktere auswählen an. Die Snowboarder gaben das Warten auf den Respawn, nachdem sie gefallen sind, an.

A.3 Begleitprotokolle

A.3.1 Arbeitsprotokoll Simon Buketits

Datum	Tätigkeit	Dauer	Tool
9/8/22	Character Controller, Local Multiplayer, VR Player	7	Unity, Rider
9/9/22	Audio System, Toon Shader and Outline	8	Unity, Rider
9/10/22	Character Controller, Local Multiplayer, VR Player	7	Unity, Rider
9/11/22	VR-Player, Grab	7	Unity, Rider
9/12/22	Interaction System, Player Fall and Stand up	7	Unity, Rider
9/15/22	Character Bugfixes	5	Unity, Rider
9/21/22	Character Selection and Editor Full Screen	7	Unity, Rider
9/29/22	Character Slide Ability and Bugfixes	6	Unity, Rider
10/13/22	Character Bugfixes	2	Unity, Rider
10/14/22	Snowman Spell, Music implemented, Ragdolls vanish	5	Unity, Rider
11/16/22	Ice Spell	4	Unity, Rider
11/17/22	Bomb Spell and Shader	6	Unity, Rider
11/23/22	Double-Ice	3	Unity, Rider
12/7/22	Wiggle Animation and Head rotating to drive direction	5	Unity, Rider
12/20/22	New Soundeffects, Game Launch Screen, Track Progress Bar	6	Unity, Rider
12/20/22	Crowntimer	2	Unity, Rider
12/21/22	Crowntime Reduce Penalty	1	Unity, Rider
1/12/23	Colliders adjusted and snowtrail, Improved Camera Dolly	4	Unity, Rider
1/13/23	Only one character can be selected (Character Selection)	3	Unity, Rider
1/18/23	Obstacle Prefabs built	2	Unity, Rider
1/20/23	Implemented Rubberbanding, Improved UI, optimised image sizes	7	Unity, Rider
1/21/23	Ice Sprite improved, Bomb Improved (Particles and Shader)	5	Unity, Rider
1/23/23	Blob Shadows	3	Unity, Rider
1/24/23	Level Selection and Respawning Button Prompt	7	Unity, Rider

Datum	**Tätigkeit**	**Dauer**	**Tool**
1/25/23	Snowflakes, Fixed VR Hand, Added Bindings to Snowboards	6	Unity, Rider
1/26/23	Ragdolls improved, Main Menu Music, New Sound Effects, New VR-Room	7	Unity, Rider
1/27/23	Button Delay Removed, Graphics Settings	4	Unity, Rider
2/1/23	Fixed Tree Stump Obstacle, Overhauled Colliders	5	Unity, Rider
2/14/23	VR-Flag	4	Unity, Rider
2/15/23	Hailstorm	6	Unity, Rider
2/16/23	Enchantment Shader, Improved Layercheck, Particles	5	Unity, Rider
2/20/23	New Player Animations, Half finished Intro Cutscene	6	Unity, Rider
3/1/23	Ingame UI, Controls UI implemented	3	Unity, Rider
3/2/23	Half finished Tutorial System	7	Unity, Rider
3/8/23	Bug fixes, Tutorial finished	6	Unity, Rider
3/9/23	Intro Cutscene finished	7	Unity, Rider
3/10/23	Improved Main Menu	4	Unity, Rider
	TOTAL	189	

A.3.2 Arbeitsprotokoll Felix Kaspar

Datum	Tätigkeit	Dauer	Tool
3/9/23	All Errors Fixed	2	Unity / Rider
3/8/23	Null Reference & Layers Fix	1	Unity / Rider
3/8/23	Model Import	0.5	Unity / Rider
3/8/23	Twinnie Feedback	0.5	Markdown
3/8/23	Juice	1	Markdown
3/8/23	VR Pipe Button	1	Unity / Rider
3/3/23	Better Settings UI	2	Unity / Rider
3/2/23	Pause UI Animations & Functions	1	Unity / Rider
3/2/23	UI Defaults	0.5	Unity / Rider
3/2/23	UI Actions	1	Unity / Rider
3/2/23	Fixed UI	1	Unity / Rider
3/2/23	Canvas Scale	1	Unity / Rider
3/2/23	VR Room	2	Unity / Rider
3/2/23	EventSystems	0.5	Unity / Rider
3/2/23	Update Tutorial.unity	1	Unity / Rider
3/2/23	PauseMenu	1	Unity / Rider
3/1/23	Center Follow	0.5	Unity / Rider
3/1/23	Ragdoll Outline	0.5	Unity / Rider
3/1/23	Follow center of players in tutorial	1	Unity / Rider
3/1/23	Fixed Crown Timer	0.5	Unity / Rider
3/1/23	Character Outline Color	1	Markdown
3/1/23	Tutorial Dolly Track	0.5	Unity / Rider
3/1/23	Tutorial Track	2	Unity / Rider
3/1/23	Better Rumble Feel	1	Unity / Rider
3/1/23	Prefabs	0.5	Unity / Rider
3/1/23	Better can respawn Feedback	1	Markdown
2/23/23	PCDontRenderOutline - is rendered by vr cam now	0.5	Unity / Rider
2/23/23	DevConsole Dont Render, Table Dont Render	0.5	Unity / Rider
2/23/23	Vibration Bug Fix	0.5	Unity / Rider
2/23/23	Outline Fix, New Table Test	1	Unity / Rider
2/23/23	Vibration on when Spell is Charged	3	Markdown
2/16/23	Particles	1	Unity / Rider
2/16/23	Detatch Particle-System on cast	0.5	Unity / Rider
2/16/23	Improved Spell Activation	0.5	Unity / Rider
2/16/23	Improved Pingu Force & Better activation for IceZone (Test)	0.5	Unity / Rider
2/16/23	Spell Cleanup	2	Unity / Rider

Datum	Tätigkeit	Dauer	Tool
2/16/23	Spell interacts	2	Markdown
2/16/23	Color	1	Unity / Rider
2/16/23	Shader Improvements & Build	2	Unity / Rider
2/16/23	Align to dolly	1.5	Unity / Rider
2/15/23	Particles	2	Unity / Rider
2/15/23	Highlights	1	Unity / Rider
2/15/23	Lighting	3	Unity / Rider
2/15/23	Room	2	Unity / Rider
2/15/23	Naming	1	Unity / Rider
2/15/23	Win Screen	2	Unity / Rider
2/3/23	Hover Highlight Fix	1	Unity / Rider
2/1/23	Squeezable	2	Markdown
1/27/23	Fixed Wrong Deaths	1	Unity / Rider
1/26/23	Icons, Death debug messages	2	Unity / Rider
1/26/23	Icons	1	Unity / Rider
1/26/23	Reverted to Timer	0.5	Unity / Rider
1/26/23	Crown Timers	1	Markdown
1/26/23	Crown Timers & New Icons	2	Unity / Rider
1/26/23	Settings	2	Unity / Rider
1/26/23	Tree Trunk fix	2	Unity / Rider
1/26/23	New Tree Trunk	1	Markdown
1/26/23	Ice Spell	0.5	Markdown
1/26/23	Better Cherry Bomb	0.5	Unity / Rider
1/26/23	Reactivated Players	1	Unity / Rider
1/26/23	New Highlighting System	3	Unity / Rider
1/26/23	Respawn Timer	1	Markdown
1/26/23	Player UI	1	Unity / Rider
1/25/23	Player Status	2	Unity / Rider
1/25/23	Hand Texture	5	Unity / Rider
1/25/23	Dead Player Vis	2	Unity / Rider
1/25/23	Hands	1	Unity / Rider
1/25/23	Hand & Player icons	2	Unity / Rider
1/25/23	Player Selection	1	Unity / Rider
1/24/23	Hand Tests	0.5	Unity / Rider
1/24/23	VR Room	0.5	Unity / Rider
1/24/23	Train	0.5	Unity / Rider
1/24/23	Deselect Unity Addon	1	Unity / Rider
1/20/23	Update PlayerPanel.prefab	1	Unity / Rider
1/20/23	Players to ScriptableObjects	0.5	Unity / Rider
1/20/23	Random Character Selection	1	Unity / Rider
1/20/23	Crown Icons	2	Unity / Rider

Datum	Tätigkeit	Dauer	Tool
1/20/23	Audio Output Detection & Automatic Setup	1	Wwise / Rider
1/19/23	Build Error Fix	0.5	Unity / Rider
1/18/23	Dispensers & Hands	0.5	Markdown
1/13/23	Force Bubble	2	Unity / Rider
1/13/23	Spell Restrictions and Feedback for Distance	3	Unity / Rider
1/12/23	Bounding Box	2	Unity / Rider
1/12/23	Cast Restriction Reasons	1	Markdown
1/12/23	Spells	1	Markdown
1/12/23	Crown Timer	2	Unity / Rider
1/12/23	Update PlayerStatusUI.cs	2	Unity / Rider
1/12/23	Player	1	Unity / Rider
1/11/23	Prefabs	1	Unity / Rider
1/11/23	Collider + Interactable Fix	2	Unity / Rider
1/11/23	Trail Fix	0.5	Unity / Rider
12/21/22	Audio	1	Wwise / Rider
12/21/22	VR Sound	1	Unity / Rider
12/21/22	LevelUI	0.5	Unity / Rider
12/21/22	Sounds	1	Unity / Rider
12/21/22	Camera	0.5	Unity / Rider
12/9/22	Snow Trail	2	Markdown
12/8/22	Player Controller	1	Unity / Rider
12/8/22	Scrolling Material	1	Unity / Rider
12/8/22	Debugger Improvements	1	Unity / Rider
12/8/22	Scrolling UV + PlayerController Debug	0.5	Unity / Rider
12/7/22	Crown Change Event	1	Unity / Rider
12/7/22	PlayerCornerIcons	1	Unity / Rider
12/7/22	PlayerGroup	1	Unity / Rider
12/7/22	Cleanup	1	Unity / Rider
12/7/22	Screen Fades	2	Unity / Rider
12/7/22	PlayerStatusUI	3	Unity / Rider
12/7/22	Game Manager Components	2	Unity / Rider
11/25/22	Cache Selected Audio Device	1	Unity / Rider
11/24/22	Multiple Audio Devices Dropdowns & Bugfixes	2	Unity / Rider
11/24/22	Flips Framedependant	2	Unity / Rider
11/24/22	Cleanup	1	Unity / Rider
11/24/22	CrownManager Fixes & BGMat Fix	1	Unity / Rider
11/24/22	Restructure	2	Unity / Rider

Datum	**Tätigkeit**	**Dauer**	**Tool**
11/24/22	Spawning Bug Fixes	1	Unity / Rider
11/23/22	Better Flip	2	Unity / Rider
11/23/22	Button	1	Unity / Rider
11/23/22	Materials	0.5	Unity / Rider
11/23/22	Juice Spell Spawner	1	Unity / Rider
11/23/22	Juice	2	Unity / Rider
11/18/22	Script & Prefab Cleanup & Organisation	3	Unity / Rider
11/18/22	Juice	1	Markdown
11/17/22	Spell Wiggle	1	Unity / Rider
11/17/22	Breitere Strecke	1	Markdown
11/17/22	Spell Anordnung	0.5	Unity / Rider
11/16/22	Mana	1	Unity / Rider
11/16/22	Cleanup	1	Unity / Rider
11/16/22	Mana Balancing	1	Markdown
11/16/22	Mana	1	Markdown
11/16/22	Mana Vis	2	Unity / Rider
11/16/22	Kessel & Ice Spell	1	Markdown
11/11/22	Audio Device Dropdown	2	Unity / Rider
11/10/22	Mana	1	Markdown
11/10/22	Mana System	2	Markdown
11/10/22	Spell Cracks	2	Unity / Rider
11/10/22	Audio	2	Wwise / Rider
11/3/22	Audio Devices	3	Wwise / Rider
10/20/22	Spells	1	Markdown
10/20/22	Button & Lights	2	Unity / Rider
10/20/22	Spells & PC-Controller	1	Unity / Rider
10/14/22	Spells & Cleanup	0.5	Unity / Rider
10/13/22	New Spell System	1	Markdown
10/13/22	Cleanup & Trigger Interactions	2	Unity / Rider
10/12/22	Interactions & Build Controls	2	Unity / Rider
10/12/22	Fixed Interactions Test	1	Unity / Rider
10/7/22	Wwise + Performance	1	Wwise / Rider
10/6/22	Wwise	1	Wwise
10/6/22	Wwise	2	Wwise
9/29/22	Rotation & grabbables	1	Markdown
9/29/22	Basic Interactables	2	Markdown
9/29/22	Rotation of VR Player	1	Unity / Rider
9/29/22	Relative Stick Rotation	1	Unity / Rider
9/8/22	Lawine and Ramp	1	Markdown
	TOTAL	198	

A.3.3 Arbeitsprotokoll Emma Wehinger

Datum	Tätigkeit	Dauer	Tool
9/6/22	Charakter Konzepte	2	Photoshop
9/7/22	Charakter Konzepte	3	Photoshop
9/8/22	Charakter Konzepte	2	Photoshop
9/15/22	Charakter Konzepte	2	Photoshop
9/16/22	Environment Konzept	2	Photoshop
9/22/22	Environment Konzept	4	Photoshop
9/23/22	VR Raum Konzept	2	Photoshop
9/29/22	VR Raum Konzept	2	Photoshop
9/29/22	Charakter Konzepte	4	Photoshop
9/30/22	VR Hände Konzept	2	Photoshop
10/5/22	Charakter Konzepte	3	Photoshop
10/6/22	Größenverhältnisse Assets	3	Photoshop
10/6/22	Menu Screen	3	Photoshop
10/7/22	Menu Screen	2	Photoshop
10/11/22	Menu Screen	2	Photoshop
10/12/22	Menu Screen	3	Photoshop
10/13/22	Menu Screen	2	Photoshop
10/14/22	Menu Screen	2	Photoshop
10/20/22	Charakter Konzepte	4	Photoshop
10/21/22	Environment Konzept	2	Photoshop
11/3/22	Plakat	6	Photoshop
11/4/22	Plakat	2	Photoshop
10-Nov	Charakter Art	4	Photoshop
11/16/22	Charakter Art	3	Photoshop
11/17/22	Charakter Art	4	Photoshop
11/18/22	Selection Menu Mockup	2	Photoshop
11/23/22	Selection Menu Mockup	3	Photoshop
11/24/22	Selection Menu Mockup	2	Photoshop
11/24/22	User Interface	4	Photoshop
11/25/22	User Interface	2	Photoshop
11/30/22	User Interface	3	Photoshop
12/1/22	User Interface	4	Photoshop
12/2/22	Main Menu Screens	2	Photoshop
12/7/22	Main Menu Screens	3	Photoshop
12/9/22	Main Menu Screens	2	Photoshop
12/14/22	Main Menu Screens	3	Photoshop
12/15/22	Main Menu Screens	4	Photoshop
12/16/22	Main Menu Screens	2	Photoshop
12/21/22	Charakter Icons	3	Photoshop

Datum	**Tätigkeit**	**Dauer**	**Tool**
1/11/23	User Interface	4	Photoshop
1/12/23	User Interface	2	Photoshop
1/13/23	Charakter Icons	2	Photoshop
1/18/23	Charakter Art	3	Photoshop
1/19/23	Charakter Art	2	Photoshop
1/20/22	Icons	2	Photoshop
1/24/23	Icons	2	Photoshop
1/25/23	Icons	3	Photoshop
1/26/23	User Interface	2	Photoshop
2/1/23	Charakter Art	3	Photoshop
2/2/23	Snowboard Texturen	6	Photoshop
2/15/23	Snowboard Texturen	3	Photoshop
2/16/23	Icons	6	Photoshop
2/21/23	Comic	2	Photoshop
2/22/23	Comic	6	Photoshop
2/23/23	Comic	6	Photoshop
3/1/23	Comic	6	Photoshop
3/2/23	Comic	6	Photoshop
3/3/23	Comic	2	Photoshop
3/8/23	Comic	6	Photoshop
3/9/23	Comic	6	Photoshop
3/10/23	Comic	2	Photoshop
3/15/23	Comic	3	Photoshop
	TOTAL	194	

A.3.4 Arbeitsprotokoll Ian Memic

Datum	Tätigkeit	Dauer	Tool
9/6/22	Charakter Textur	1	Blender
9/6/22	ersten Charakter modelliert	2	Blender
9/6/22	Kronen-Assets modelliert	0.5	Blender
9/7/22	VR-Szene modelliert	6	Blender
9/8/22	Charakter Rigging	2	Blender
9/15/22	Charakter modelliert	5	Blender
9/15/22	Charakter texturiert	1	Blender/Photoshop
9/16/22	Asset optimierung	2	Blender
9/22/22	Candy-Assets modelliert	5	Blender
9/22/22	Assetes texturiert	2	Blender
9/23/22	Assets modelliert	4	Blender
9/29/22	Assets texturiert	2	Blender/Photoshop
9/29/22	Assets modelliert/texturiert	4	Blender
9/29/22	Jumppad geriggt	0.5	Blender
9/30/22	Charakter modelliert	2	Blender
10/5/22	Charakter texturiert	2	Photoshop
10/5/22	VR-Assets modelliert	3	Blender
10/6/22	VR-Assets modelliert	6	Blender
10/6/22	VR-Assets modelliert	6	Blender
10/7/22	VR-Assets texturiert	2	Blender/Photoshop
10/7/22	VR-Assets modelliert	4	Blender
10/11/22	VR-Assets texturiert	2	Blender/Photoshop
10/11/22	VR-Assets modelliert	2	Blender/Photoshop
10/12/22	VR-Assets modelliert	5	Blender
10/12/22	VR-Assets texturiert	1	Blender/Photoshop
10/13/22	VR-Assets modelliert	5	Blender
10/14/22	VR-Assets modelliert	5	Blender
10/20/22	VR-Strecke modelliert	5	Blender
10/21/22	VR-Strecke modelliert	3	Blender
11/3/22	VR-Assets modelliert	6	Blender
11/4/22	VR-Assets texturiert	2	Blender/Photoshop
10-Nov	VR-Assets texturiert	3	Blender/Photoshop
11/16/22	VR-Szene modelliert	6	Blender
11/17/22	VR-Szene modelliert	6	Blender
12/12/22	Partikel-System	2	Unity
12/12/22	VR-Strecke modelliert	4	Blender
12/13/22	VR-Strecke modelliert	6	Blender
1/18/23	VR-Strecke modelliert	6	Blender
1/19/23	VR-Hände modelliert	4	Blender

Datum	**Tätigkeit**	**Dauer**	**Tool**
1/24/23	VR-Szene modelliert	4	Blender
1/25/23	VR-Hände modelliert	6	Blender
1/26/23	VR-Szene modelliert	6	Blender
2/1/23	Pinguin-Asset modelliert	4	Blender
2/1/23	VR-Szene modelliert	2	Blender
2/2/23	Pinguin-Asset texturiert	2	Blender/Photoshop
2/15/23	Kerzen-Assets modelliert	1	Blender
2/15/23	Kessel-Asset modelliert	2	Blender
2/15/23	Kerzen-Assets modelliert	1	Blender
2/15/23	Kristallkugel-Asset modelliert	2	Blender
2/23/23	VR-Assets modelliert	6	Blender
3/1/23	Fass-Asset modelliert	3	Blender
3/1/23	VR-Assets modelliert	3	Blender
3/1/23	VR-Assets texturiert	2	Blender
3/2/23	VR-Strecke modelliert	2	Blender
3/2/23	Strecken-Assets modelliert	1	Blender
3/2/23	VR-Assets texturiert	3	Blender
3/2/23	Assets exportieren	1	Blender
3/8/23	VR-Assets modelliert	5	Blender
3/9/23	VR-Assets modelliert	4	Blender
3/9/23	Item-Assets modelliert	1	Blender
3/15/23	VR-Assets modelliert	2	Blender
	TOTAL	198	

Anhang B

Kooperationsvereinbarung

KOOPERATIONSVEREINBARUNG

zwischen

1. CYBERTIME GmbH
2. A-1060 Wien, Mariahilfer Straße 113
(Name und Adresse des Unternehmens)

(in der Folge **„der Projektpartner, die Projektpartnerin"**)

und

3. Simon Buketits

Felix Kaspar

Emma Wehinger

Ian Memic

(Namen der Schüler/Schülerinnen)

(in der Folge **„das Projektteam")**

PRÄAMBEL

Das Projektteam und der Projektpartner/die Projektpartnerin beabsichtigen gemäß der Prüfungsordnung BMHS, BGBl II Nr. 177/2012 i.d.g.F., die Planung und Durchführung eines Diplomprojekte mit dem Titel:

.....

welches die Erstellung von Arbeitsergebnissen laut beiliegender Projektbeschreibung zum Gegenstand hat.

Durch die Zusammenarbeit soll insbesondere den Mitgliedern des Projektteams die Möglichkeit eingeräumt werden, im Rahmen ihrer schulischen Ausbildung bei der Durchführung eines Diplomprojektes an die Verhältnisse im technischen Berufsleben herangeführt zu werden, um dabei die in der Schule erworbenen theoretischen Kenntnisse und Fähigkeiten in der Praxis anzuwenden bzw. zu erweitern. Hingewiesen wird in diesem Zusammenhang auf den unentgeltlichen Charakter dieser Vereinbarung.

§1
Gegenstand

Gegenstand ist die Erstellung von Arbeitsergebnissen zum Thema des Diplomprojekts gemäß beiliegender Projektbeschreibung. Der Projektpartner / die Projektpartnerin wird jedoch darauf hingewiesen, dass es sich um ein Projekt im Zusammenhang mit der schulischen Ausbildung handelt und daher jede Haftung des Projektteams, insbesondere in Hinsicht auf die Unentgeltlichkeit des Vertrages, ausgeschlossen ist.

§2
Laufzeit

Die vorliegende Kooperation tritt am 20.9.2022 in Kraft und wird bis zum schulrechtlich verordneten Termin am 31.3.2023 abgeschlossen.

§ 3
Rechte und Pflichten des Projektteams

Das Projektteam verpflichtet sich, die im Gegenstand genannten Arbeiten sorgfältig und unter möglichster Schonung der Interessen des Projektpartners / der Projektpartnerin durzuführen.
Das Projektteam verpflichtet sich zur Geheimhaltung aller ihm zur Kenntnis gelangenden Geschäfts- und Betriebsgeheimnisse gegenüber Dritten. Ausnahmen siehe §5 Einsicht und Präsentation.

§4
Rechte und Pflichten des Projektpartners / der Projektpartnerin

Der Projektpartner / die Projektpartnerin verpflichtet sich, das Projektteam nach Kräften zu unterstützen und dem Projektteam folgende Hilfsmittel zeitgerecht und für die Dauer des Diplomprojektes kostenfrei zur Verfügung zu stellen:

- BERATUNG
- HILFESTELLUNGEN
- .....

Sofern der Projektpartner / die Projektpartnerin dem Projektteam urheberrechtlich geschütztes Material oder Daten zur Verfügung stellt, stellt der Projektpartner / die Projektpartnerin sicher, dass dieses Material frei von Rechten Dritter ist. Der Projektpartner / die Projektpartnerin hält das Projektteam diesbezüglich schad - und klaglos.
Sollte das Projektteam im Rahmen dieser Kooperationsvereinbarung ein Werk schaffen, dem Schutz im Sinne des Urheberrechtsgesetzes zukommt, hat der Projektpartner / die Projektpartnerin uneingeschränkte Nutzungs- und Verwertungsrechte. Das Projektteam verpflichtet sich in diesem Zusammenhang keine Ansprüche bezüglich einer Vergütung zu erheben.

§5
Einsicht und Präsentation

Da die Tätigkeit des Projektteams auch Inhalt bzw. Grundlage der an der Schule HTBLuVA Wien V, Spengergasse 20 zu erstellenden Diplomarbeit ist, berechtigt der Projektpartner / die Projektpartnerin die zuständigen Organe des Bundes zur Einsicht und Kontrolle, um die Aufgaben gem. Prüfungsordnung BMHS, SchUG bzw. SchUG BKV zu erfüllen. Das Projektteam ist auch berechtigt, Ergebnisse der Diplomarbeit im Rahmen des Schulunterrichts und bei Schul- und schulbezogenen Veranstaltungen, sowie bei der Präsentation und Diskussion der Diplomarbeit zu verwenden.

§6
Änderungen

Änderungen dieser Vereinbarung bedürfen der Schriftform. Sollte ein Schüler / eine Schülerin, der / die Mitglied des Projektteams ist, während der Laufzeit dieser Vereinbarung aus der HTBLuVA Wien V, Spengergasse 20 ausscheiden (Abmeldung vom Schulbesuch), bleibt die Kooperationsvereinbarung für die verbleibenden Unterzeichner, mit Rücksichtnahme auf eine etwaige Reduktion des Projektumfanges, aufrecht.

Wien, 21.09.2022

.....	.....
(Ort, Datum)	(für das Projektteam)
WIEN, 20.9.2022	cybertime GMBH Mariahilfer Straße 113 / 9 1060 Wien tel +43-1-718 55 00-50 fax +43-1-718 55 00-55 www.cybertime.at office@cybertime.at
.....	.....
(Ort, Datum)	(für den/die Projektpartner/in)

Anhang C

Inhalt des Datenträgers

C.1 Arbeit

Pfad: /Diplomarbeit

LaTex/	LaTex Projekt
_DaBa.pdf	Diplomarbeit (Gesamtdokument)

C.2 Projektdateien

Pfad: /Game

Build/	Windows Build des Spieles
Projekt/	Unity + Wwise Projekt

C.3 Game Design

Pfad: /GameDesign

Concepts/	Konzept Dateien
Playtests/	Ergebnisse der Playtests

C.4 Presskit

Pfad: /Presskit

Logos/	Logos des Spieles
Screenshots/	Screenshots des Spieles
ControllerLayout.pdf . .	Controller Layout des Spieles
Poster.png	Poster des Spieles
Presskit.pdf	Ein PDF mit Infos zum Spiel
Trailer.mp4	Trailer des Spieles

TRICKS
'N'
TREATS

A BRILLIANT IDEA!
BLA BLA
BLA BLA BLA
BLA BLA
BLA BLA
BLA BLA
BLA

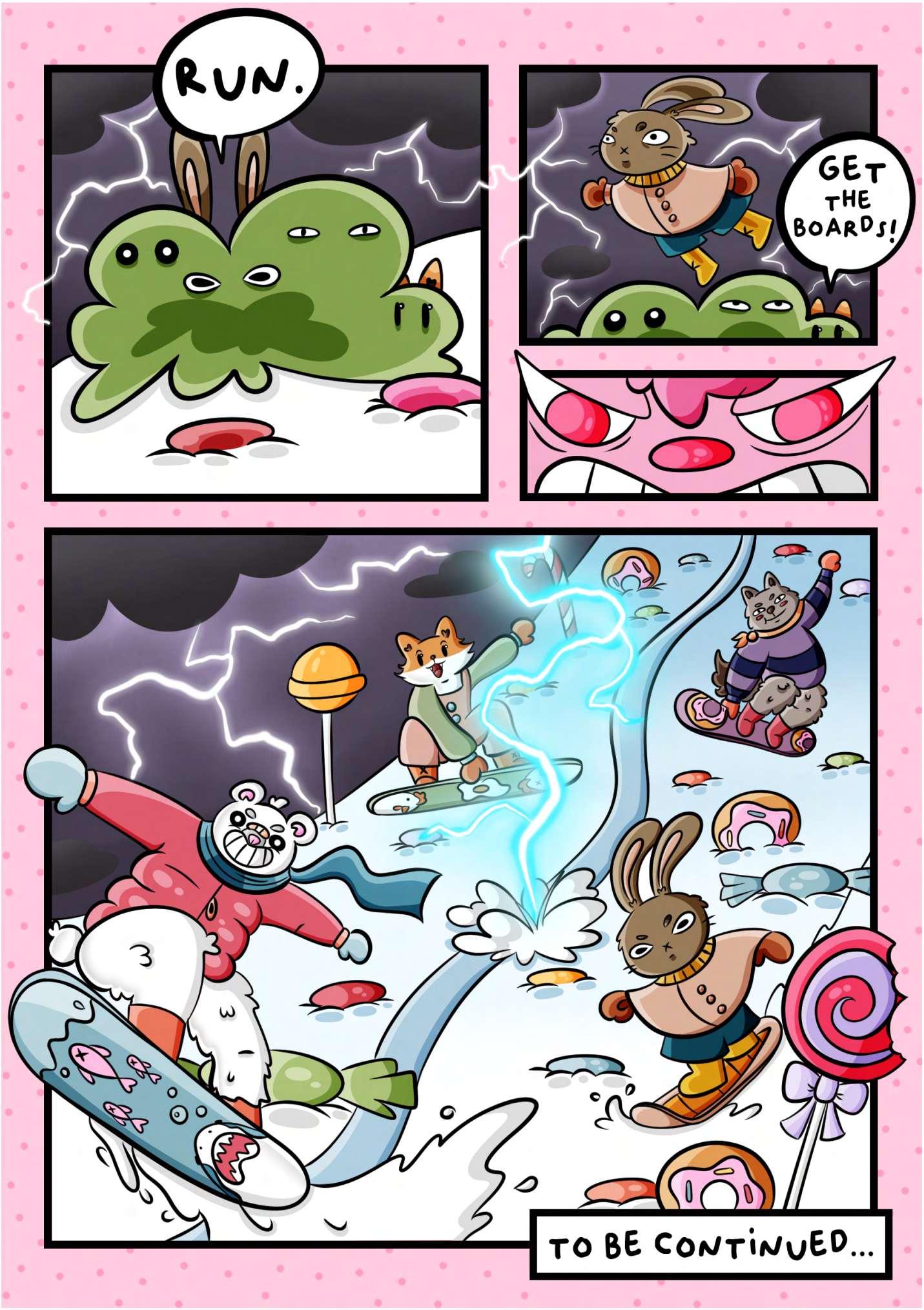
HURRY UP!
ON THEIR WAY...

THIS WAY!
?
?
?
LAND AHOY!
HUNGRY...
A SHORT BREAK...

FOOD BREAK!
MOVE YOUR ASS, NOW!
IT'S BEAUTIFUL.
THEY FINALLY ARRIVED...

THIS WAY
THIS IS HEAVEN!
WE MADE IT!
CLIMBING THE MOUNTAIN...

READY FOR PRANK BUZZING!
HELLO? IS SOMEBODY THERE???
HE IS COMING...
DUMB ASS ANIMALS! ARGH...


RUN.
GET THE BOARDS!
TO BE CONTINUED...

Literatur

[1] *Steam Hardware Survey*. eng. 2022. URL: https://web.archive.org/web/20221102113614/https://store.steampowered.com/hwsurvey/Steam-Hardware-Software-Survey-Welcome-to-Steam (siehe S. 4).

[2] *Keep Talking and Nobody Explodes Steam Store Page*. eng. 2022. URL: https://store.steampowered.com/app/341800/Keep_Talking_and_Nobody_Explodes/ (siehe S. 5).

[3] *Davigo Steam Store Page*. eng. 2022. URL: https://store.steampowered.com/app/1116540/DAVIGO/ (siehe S. 5).

[4] *Carly and the Reaperman Steam Store Page*. eng. 2022. URL: https://store.steampowered.com/app/547480/Carly_and_the_Reaperman__Escape_from_the_Underworld/ (siehe S. 5).

[5] Ben Lawson. *Motion Sickness Symptomatology and Origins*. Sep. 2014, S. 531–600 (siehe S. 6).

[6] Andreas Sieß, Sandra Beuck und Matthias Wölfel. *Virtual Reality – Quo Vadis? How to Address the Complete Audience of an Emerging Technology (Full Paper)*. eng. Sep. 2017 (siehe S. 6).

[7] *Is Your Frame Rate Affecting Your VR Experience?* eng. 2022. URL: https://web.archive.org/web/20221129092545/https://vr.linde.com/2022/10/06/is-your-frame-rate-affecting-your-vr-experience/ (siehe S. 7).

[8] *Level of Detail for Meshes*. eng. 2022. URL: https://docs.unity3d.com/Manual/LevelOfDetail.html (siehe S. 8).

[9] *Culling Explanation*. eng. 2022. URL: https://docs.cryengine.com/display/SDKDOC4/Culling+Explained (siehe S. 9, 10).

[10] *Culling Image*. eng. 2022. URL: https://techpubs.jurassic.nl/manuals/nt/developer/Optimizer_PG/sgi_html/ch05.html (siehe S. 10).

[11] *Draw Call Batching Unity Docs*. eng. 2017. URL: https://docs.unity3d.com/Manual/DrawCallBatching.html (siehe S. 10, 11).

[12] *Object Pooling Unity*. eng. 2021. URL: https://learn.unity.com/tutorial/introduction-to-object-pooling# (siehe S. 11).

[13] C. Dickinson. *Unity 5 Game Optimization*. Community experience distilled. Packt Publishing, 2015. URL: https://books.google.at/books?id=1ngujwEACAAJ (siehe S. 12).

[14] *Static and Dynamic Collider*. eng. 2022. URL: https://docs.unity3d.com/Manual/CollidersOverview.html (siehe S. 12, 13).

[15] *Oxford English dictionary*. eng. 2022 (siehe S. 22, 25).

[16] Bitcom. „Fact Sheet: Augmented Reality (AR) and Virtual Reality (VR) 2021/22“. In: *Bitkom* (2021). URL: https://www.bitkom.org/sites/default/files/2021-10/211005_ct_fact-sheet_ar-vr.pdf (siehe S. 22).

[17] John Aliprantis u. a. „Natural Interaction in Augmented Reality Context.“ In: *VIPERCIRCDL*. Bd. 2320. CEUR Workshop Proceedings. CEUR-WS.org, 2019 (siehe S. 22).

[18] *Weighted Companion Cube der von der Portalgun gehalten wird. Aus dem Spiel Portal*. GamePro. URL: https://www.gamepro.de/galerien/half_life_2_portal,52501.html (siehe S. 23).

[19] *Weighted Companion Cube der von der Portalgun gehalten wird. Aus dem Spiel Portal*. PCGAMER. URL: https://www.pcgamer.com/create-graffiti-wear-a-traffic-cone-hat-and-8-other-fun-things-to-do-in-the-half-life-alyx-intro/ (siehe S. 23).

[20] Jesse Schell. *The Art of Game Design: A book of lenses*. Taylor & Francis, 2008 (siehe S. 23, 27–30, 33, 34).

[21] *R/localmultiplayergames - shared screen or split screen. what do you prefer?* Feb. 2021. URL: https://www.reddit.com/r/localmultiplayergames/comments/laud53/shared%5C%5Fscreen%5C%5For%5C%5Fsplit%5C%5Fscreen%5C%5Fwhat%5C%5Fdo%5C%5Fyou%5C%5Fprefer/ (siehe S. 24).

[22] Bezalel Peleg und Peter Sudhölter. *Introduction to the theory of cooperative games*. Theory and decision library. Boston, Mass. [u.a.]: Kluwer Acad. Publ., 2003 (siehe S. 24).

[23] José Rocha, Samuel Mascarenhas und Rui Prada. *Game Mechanics for Cooperative Games*. Jan. 2008 (siehe S. 25).

[24] Philipp Beau und Sander Bakkes. „Automated game balancing of asymmetric video games“. In: *2016 IEEE Conference on Computational Intelligence and Games (CIG)*. 2016 (siehe S. 25).

[25] Alan McKee. „What Is Fun?“ In: *FUN!: What Entertainment Tells Us About Living a Good Life*. London: Palgrave Macmillan UK, 2016 (siehe S. 26).

[26] Raph Koster. *A Theory of Fun for Game Design*. Scottsdale, Arizona: Paraglyph Press, 2005 (siehe S. 26, 30).

[27] *Competition vs. Cooperation in Games*. 2022. URL: https://www.ab-lab.org/competition-vs-cooperation.html (siehe S. 27).

[28] Linda K Kaye und Jo Bryce. „Putting the fun factor into gaming: The influence of social contexts on the experiences of playing videogames". English. In: *International Journal of Internet Science* 7.1 (2012) (siehe S. 27).

[29] Harry Reis, Stephanie O'Keefe und Richard Lane. „Fun Is More Fun When Others Are Involved". In: *The Journal of Positive Psychology* 12 (Aug. 2016) (siehe S. 27).

[30] Robin Hunicke, Marc LeBlanc und Robert Zubek. „MDA: A Formal Approach to Game Design and Game Research". In: *Challenges in Game Artificial Intelligence — Papers from the 2004 AAAI Workshop*. Hrsg. von Dan Fu, Stottler Henke und Jeff Orkin. WS-04-04. Menlo Park, CA: American Association for Artificial Intelligence, 2004. URL: http://www.aaai.org/Library/Workshops/2004/ws04-04-001.php (siehe S. 28).

[31] T. Fullerton und an O'Reilly Media Company Safari. *Game Design Workshop: A Playcentric Approach to Creating Innovative Games*. CRC P., 2014 (siehe S. 29, 38, 40, 42).

[32] *Pacing - The Level Design Book*. 2023. URL: https://book.leveldesignbook.com/process/preproduction/pacing (siehe S. 30).

[33] Mihaly Csikszentmihalyi. *Flow: the psychology of optimal experience*. Harper Collins, 1990 (siehe S. 30).

[34] *Acron: Attack of the Squirrels! Steam Reviews*. 2023. URL: https://steamcommunity.com/app/1094870/reviews?filterLanguage=default%5C&p=1%5C&browsefilter=mostrecent (siehe S. 32).

[35] Game Developer. *User research: Why players love game that makes them frustrated*. Aug. 2021. URL: https://www.gamedeveloper.com/business/user-research-why-players-love-game-that-makes-them-frustrated (siehe S. 38).

[36] P. Mollica. *Special Subjects: Basic Color Theory: An introduction to color for beginning artists*. How to Draw & Paint. Walter Foster Publishing, 2018. URL: https://books.google.at/books?id=O9V7DwAAQBAJ (siehe S. 46–52).

[37] *Farben*. ger. 2023. URL: https://www.leifiphysik.de/optik/farben#:~:text=Spektralfarben,weiter%20in%20andere%20Farben%20zerlegen. (siehe S. 46).

[38] J. Day. *Line Color Form: The Language of Art and Design*. Allworth, 2013. URL: https://books.google.at/books?id=qFeHCgAAQBAJ (siehe S. 47–55).

[39] C. Solarski. *Drawing Basics and Video Game Art: Classic to Cutting-Edge Art Techniques for Winning Video Game Design*. Clarkson Potter/Ten Speed, 2012. URL: https://books.google.at/books?id=ZcfFQXSsWNEC (siehe S. 47–49, 51–57, 59–63).

[40] *CMYK und RGB*. ger. URL: https://www.salierdruck.de/rgb-cmyk (siehe S. 48, 49).

[41] *Subtraktive Farbmsichung*. ger. 2023. URL: https://www.leifiphysik.de/optik/farben/grundwissen/subtraktive-farbmischung (siehe S. 48).

[42] *Additive Farbmsichung*. ger. 2023. URL: https://www.leifiphysik.de/optik/farben/grundwissen/additive-farbmischung (siehe S. 49).

[43] *Farbenpsychologie*. ger. URL: https://www.maler-michels.de/wohnraumdesign/farbpsychologie/#:~:text=Die%20Wirkung%20der%20Farben,Rot%2C%20Gelb%20und%20Orange%20anregend. (siehe S. 52).

[44] Mark JP Wolf. *The video game explosion: a history from PONG to Playstation and beyond*. ABC-CLIO, 2008 (siehe S. 58).

[45] *Farbkontraste in Videospielen*. ger. URL: https://docplayer.org/74652557-Die-sieben-farbkontraste-im-gaming.html (siehe S. 58–60, 62).

[46] *Outlast*. eng. 2014. URL: https://redbarrelsgames.com/games/outlast/ (siehe S. 58, 59).

[47] *Farbkontraste*. ger. 2001. URL: http://www.tomatoco.de/valbers/kontrast1.html (siehe S. 59, 60).

[48] *Journey*. ger. 2012. URL: https://www.playstation.com/de-at/games/journey/ (siehe S. 62).

[49] *Super Mario Galaxy*. ger. 2007. URL: https://www.nintendo.at/Spiele/Wii/Super-Mario-Galaxy-283322.html (siehe S. 62).

[50] Mohan L. Kolhe u. a. *Advances in Data and Information Sciences - Proceedings of ICDIS 2019*. Singapore: Springer Nature, 2020 (siehe S. 74).

[51] Dr. Davide Aversa und Chris Dickinson. *Unity Game Optimization - Enhance and extend the performance of all aspects of your Unity games, 3rd Edition*. Birmingham: Packt Publishing Ltd, 2019 (siehe S. 75, 77, 81, 84, 85, 87, 103).

[52] *A Guide to Building a 3D Game Character*. eng. 2017. URL: https://www.theseus.fi/bitstream/handle/10024/133905/Heikkila_Elisabet.pdf (siehe S. 75).

[53] *Virtual reality sickness 'tackled with field of view trick'*. eng. 2016. URL: https://www.bbc.com/news/technology-36550369 (siehe S. 77).

[54] Rob Bateman und Richard Olsson. „The Essential Guide to 3D in Flash“. In: (Jan. 2010) (siehe S. 77, 78, 88).

[55] *Geometry Topology*. eng. 2020. URL: https://cs184.eecs.berkeley.edu/sp19/lecture/8-15/meshes-and-geometry-processing (siehe S. 78).

[56] Nicky Webster. „High poly to low poly workflows for real-time rendering“. In: *Journal of Visual Communication in Medicine* 40 (Jan. 2017), S. 40–47 (siehe S. 78, 91, 92).

[57] *CD Projekt refunded 30K copies of Cyberpunk 2077 out of 13M*. eng. 2021. URL: https://www.polygon.com/2021/4/22/22398340/cd-projekt-red-april-2021-investors-call-refund-details-costs (siehe S. 83).

[58] Ana Gomes Sarmento Da Fonseca. „3D modeling pipeline for games : work methods for low poly models with hand painted textures“. In: 2018 (siehe S. 83, 84, 86).

[59] *Grundlegende Prinzipien Der Low-Poly-Modellierung*. ger. 2021. URL: https://3dcoat.com/de/articles/article/basic-principles-of-low-poly-modeling/ (siehe S. 84).

[60] *Textures and Samplers in Metal*. eng. 2014. URL: https://metalbyexample.com/textures-and-samplers/ (siehe S. 85).

[61] John M. Blain. *The Complete Guide to Blender Graphics - Computer Modeling & Animation, Fourth Edition*. Boca Raton, Fla: CRC Press, 2017 (siehe S. 85).

[62] Luke Ahearn. *3D Game Textures - Create Professional Game Art Using Photoshop*. Chapman und Hall/CRC, London: CRC Press, Taylor & Francis Group, 2016 (siehe S. 85, 91).

[63] *Different maps in PBR Textures*. eng. 2023. URL: https://www.a23d.co/blog/different-maps-in-pbr-textures/ (siehe S. 86).

[64] *Diffrent Types of Shading*. chn. 2015. URL: https://blog.csdn.net/silangquan/article/details/45629299 (siehe S. 90).

[65] L. Ahearn. *3D Game Textures: Create Professional Game Art Using Photoshop*. CRC Press, 2016. URL: https://books.google.li/books?id=qmkNDgAAQBAJ (siehe S. 91).

[66] *What's the difference between Bump Maps, Normal Maps and Height Maps?* eng. 2023. URL: https://docs.unity3d.com/Manual/StandardShaderMaterialParameterNormalMap.html (siehe S. 92).

Abbildungsverzeichnis

2.1 Ergebnisse des Steam Hardware Surveys aus dem Jahr 2022[1] 4
2.2 VR-Riese und PC-Charakter in dem Spiel „Davigo“[3] 5

3.1 Ein Mesh mit zwei verschiedenen Detailgraden[8] 8
3.2 Sehpyramide ohne Culling, mit Frustrum Culling und mit Frustrum und Occlusion Culling[10] 10

4.1 Editor View eines unfertigen Levels, Die rote Kapsel stellt den VR-Charakter dar 14
4.2 Korrekte Collider 17
4.3 Top Down Layout des VR-Raumes mit dem Berg in der Mitte, blaue Zone stellt rotierbaren Bereich dar 18

6.1 Grab in Portal[18] vs Alyx[19] 23

7.1 MDA Framework[30] 28
7.2 Flowchannel - The Art of Game Design[20] 30

10.1 Screenshot einer Situation im Spiel aus der Sicht der PC-Spielenden 41

11.1 Farbprisma[36] 47
11.2 Farbrad[36] 48
11.3 Farbrad in warme und kalte Farben unterteilt[36] 50
11.4 Farbrelativität zwischen zwei Farben[36] 51
11.5 Skala der Formen[39] 53
11.6 Kompositionslinien in einem Werk[39] 55
11.7 Character-Centric-Design[39] 56

12.1 “Outlast“ In-Game Screenshot[39] 59
12.2 “Journey“ In-Game Screenshot[39] 60
12.3 Video-Game Character-Shapes[39] 61
12.4 “Journey“ In-Game Screenshot [48] 62
12.5 “Super Mario Galaxy“ In-Game Screenshot[39] 63

13.1 Character-Design der Snowboarder 65
13.2 Artwork des Magiers 67
13.3 Menu-Screen-Mockup 68
13.4 Selection-Menu-Mockup 69
13.5 Environment-Konzept 70
13.6 Start-Menu-Screen 71

14.1 Rendering Beispiel: 2 Kameras in Unity 76
14.2 VR-Sicht mit zwei Linsen 77
14.3 Unterschied zwischen Topologie und Geometrie[55] 78
14.4 Visuelle Darstellung von Vertices 79
14.5 Visuelle Darstellung von Edges 79
14.6 Visuelle Darstellung von Faces 80
14.7 Tri, Quad & N-Gon 80
14.8 Shading-Probleme bei N-Gons 81
14.9 Spiel Screenshot beim Launch 82
14.10Mesh Resolution eines Assets 84
14.11Textur eines Charakters 85
14.12Unterschiedliche Arten von Texturen 86
14.13Beispiel für eine UV-Map 87
14.14Einfaches Billboard 88
14.15Verschiedene Arten von Shading: Flat - Gouraud - Phong . . 90
14.16Unterschied zwischen Bump-Maps und Normal-Maps 92

15.1 Verschiedene Auflösungen des selben Meshes 95

16.1 Vergleich: 1200 Faces vs 6200 Faces 99
16.2 Unötige Faces 100
16.3 Vergleich: Flat Shading vs. Smooth Shading 101
16.4 Textur Palette und Charaktere 102