

3D-Modeling mit Blender: Einen Charakter zum Leben erwecken



Cansu Gül, G4A
Layla Hamad, G4A

Maturaarbeit Alte Kantonsschule Aarau
Oktober 2019
Betreut von Martina Vazquez

Abstract

Die vorliegende Maturaarbeit gibt einen Überblick über die Software Blender 2.8, welche unter anderem zur Erstellung von 3D-Modellen verwendet wird. Die Software Blender besitzt unterschiedliche Funktionen, welche im Zusammenhang mit Gaming, 3D-Modellierung, Rigging, Character Animation und Rendering stehen [1].

Das Ziel dieser Arbeit war es, in einer bestimmten Zeitspanne durch diverse Tutorials und Literatur selbstständig von Anfängern zu Experten dieses Programms zu werden. Die theoretischen Grundlagen des 3D-Modeling und die Funktionen des Programms werden genau untersucht, vertieft und detailliert erläutert.

Das Ergebnis dieser Arbeit sind ein eigens erstelltes 3D-Modell und ein Hotkey-Sheet. Der Prozess ging vom Entwurf eines eigenständigen Charakters bis hin zur Erstellung des 3D-Modells, welches aus einem Polygon-Mesh besteht, welches mit Hilfe eines Rigs posiert werden kann. Zusätzlich wurde ein Cel Shader erstellt, dessen Funktionsweise und Verwendung in dieser Arbeit erklärt werden. Der Bereich der 3D-Modellierung ist relativ jung und durch seine verbreitete Verwendung in der Film- und Videospielbranche zugleich topaktuell, daher ist diese Maturaarbeit sowohl für die Studierenden der Alten Kanti als auch für Interessierte aus den Bereichen Informatik, Mathematik und Bildnerisches Gestalten spannend.

Inhaltsverzeichnis

1	Einleitung.....	1
1.1	Vorwort.....	1
1.2	Ziele der Arbeit.....	2
2	Grundlagen und Theorie	3
2.1	Blender.....	3
2.1.1	Die Entwicklung	3
2.1.2	Programmiersprache und Skriptsprache	4
2.1.3	Blender 2.8	5
2.1.4	User Interface von Blender 2.8	6
2.2	Shader.....	10
2.2.1	Funktionsweise eines Shaders	10
2.2.2	Phong Reflection Model	12
2.2.3	Cel Shading	16
2.2.4	Nodes in Blender	17
2.3	Low Poly.....	20
3	Erstellungsprozess des Charakters.....	22
3.1	Charakterdesign.....	22
3.2	Erstellung des 3D-Modells.....	23
3.3	Erstellung eines Cel Shader in Blender	27
3.3.1	Zusammenhänge der Nodes	27
3.3.2	Pseudocode des Cel Shader	29
4	Resultate des Erstellungsprozesses.....	31
5	Diskussion	34
6	Schluss	36
7	Glossar	I
8	Literaturverzeichnis	V
9	Abbildungsverzeichnis	XI
	Anhang.....	XIII

1 Einleitung

1.1 Vorwort

Vor Ihnen liegt die Maturaarbeit „3D-Modeling mit Blender: Einen Charakter zum Leben erwecken“. In dieser wurde mit Hilfe des Programms Blender 2.8 ein 3D-Modell erstellt, dabei wurden theoretische Grundlagen der 3D-Modellierung angeschaut und an dem Modell angewendet. Auf dieses Thema sind wir durch unser Interesse, eine neue Welt kennenzulernen, gestossen. Wir hatten viel über das Programm Blender zur 3D-Modellierung gehört und daraus folgte, dass wir das Programm genauer unter die Lupe nehmen wollten. Die Faszination lag darin, dass alles, was in der Wirklichkeit existiert oder vielleicht existieren könnte, in der virtuellen Welt dargestellt werden kann.

An dieser Stelle möchten wir uns bei unserer Betreuerin Frau Martina Vazquez bedanken. Zusammen mit ihr haben wir die Fragestellung für unsere Arbeit entwickelt, und auch dank ihren wertvollen Ratschlägen gelang es uns, eine klare Struktur unserer Arbeit zu schaffen. Ebenso gebührt unser Dank dem SAE-Institut, welches unterschiedliche Kurse zur Verfügung stellt. Durch die besuchten Kurse konnten wir unser Know-how, welches wir durch YouTube-Tutorials erlangt haben, vertiefen. Besonders die Videos und Anleitungen zu Blender, Blender Guru, Cherylynn Lima und CG Boost boten uns gute Einblicke und Inspiration.

Zuletzt danken wir Tobias Bollinger, dessen Ratschläge stets hilfreich und nützlich waren, und unseren Freunden und Familien, die uns während dem Verfassen dieser Arbeit unterstützt haben.

Die Schnittstelle zwischen Informatik und Kunst ermöglichte uns nicht nur, unserer Kreativität freien Lauf zu lassen, sondern liess uns eine Facette der digitalen Welt kennenlernen. Die Kenntnis der 3D-Modellierung eröffnet viele Möglichkeiten, denn die erstellten Objekte können in vielen unterschiedlichen Bereichen wie Animation und Videospiele Anwendung finden. Eine Fähigkeit zu erlernen, von welcher wir zu einem späteren Zeitpunkt profitieren können, spornte uns dazu an, in die Welt der Game Artists und 3D-Designer einzutauchen.

Das erstellte 3D-Modell, sowie das Hotkey-Sheet sind unter <https://github.com/laylahamad/Maturaarbeit-3D-Modeling-mit-Blender-Einen-Charakter-zum-Leben-erwecken> zu finden. Gewisse in der Arbeit verwendete Fachbegriffe werden im Glossar erklärt.

1.2 Ziele der Arbeit

In dieser Maturaarbeit wird ein Programm namens Blender vorgestellt und verwendet, wobei es sich konkret um dessen Version Blender 2.8 handelt. Heutzutage existieren zahlreiche Programme zur 3D-Modellierung, doch die Wahl fiel auf Blender, da dieses zu den bekanntesten 3D-Modellierungsprogrammen gehört, viele verschiedene Funktionen besitzt sowie kostenlos verfügbar ist. Aufgrund seiner Beliebtheit haben 3D-Artists, aber auch die Blender Foundation selbst, zahlreiche Tutorials erstellt, welche uns das Erlernen des Programms ermöglicht haben.

Trotz der künstlerischen Anteile wurde diese Arbeit im Bereich der Informatik verfasst, da technische und mathematische Aspekte des 3D-Modeling umfangreich behandelt werden. Ursprünglich war auch ein 3D-Druck vorgesehen, dieses Ziel fiel aufgrund der damit verbundenen Kosten allerdings weg. Ursprünglich sollte zudem die Version 2.8 im Vergleich mit älteren Versionen von Blender dargestellt werden. Auch auf diesen Punkt wurde jedoch verzichtet und die Ziele wurden schliesslich wie folgt formuliert:

Die Hauptziele dieser Arbeit waren, den Umgang mit Blender zu erlernen und so die Fähigkeiten des 3D-Modeling zu erlangen. Unter Anwendung dieser Fähigkeiten sollte ein Hotkey-Sheet und ein komplexes 3D-Modell erstellt werden, welches im Vergleich mit anderen Modellen aus neueren Videospielen als Low-Poly gilt und sichtbare Ecken besitzt, durch Bones beweglich gemacht wird und so in einem Spiel implementiert werden könnte. Das Modell weist keinerlei Lücken im Mesh und bei Bewegungen, die ein normaler Mensch machen kann, treten keine schwerwiegenden Verzerrungen auf.

Ein weiteres Ziel war, einen Shader zu erstellen, welcher die Merkmale eines Cel Shader aufweist und in Blender auf das Objekt angewendet werden kann, und seine Funktionsweise zu erläutern.

2 Grundlagen und Theorie

2.1.1 Die Entwicklung

Blender ist eine Open-Source-3D-Software, welche im Jahr 1998 als ursprünglich firmeninternes Programm des Animationsstudios NeoGeo erschien [1]. Eine kommerzielle Weiterentwicklung schien nach dem Scheitern des Unternehmens Not a Number Technologies im Jahre 2002 nicht realistisch, was im selben Jahr zur Gründung der Stiftung Blender Foundation führte. Diese setzte sich nach einer Einmalzahlung von 100'000 Euro für die Freigabe des Quellcodes zum Ziel, Blender als Open-Source-Projekt weiterzuführen. Der finanzielle Unterhalt dieser Software wird durch Spenden gewährleistet. [2] [3] [4] [5]



Abbildung 1: Das Blender-Logo [2]

Mit Blender lassen sich Körper erstellen und texturieren und anschliessend rendern oder animieren. Durch Motion Capture, Simulationen und ein integriertes Videoschnittprogramm wird dem Benutzer das Kreieren von Animationen direkt in Blender ermöglicht [3] [5]. Trotz ihres Funktionsumfangs ist die Software im komprimierten Zustand mit rund 100 MB relativ klein und läuft ohne Installation auf verschiedenen Betriebssystemen. [3] [6]

Blender wird von professionellen und ehrenamtlichen Programmierern weiterentwickelt. Eine wichtige Rolle spielen dabei Projekte, in welchen Künstler und Programmierer gemeinsam ein Produkt, zum Beispiel ein Videospiel oder einen Film, erstellen und durch die nötigen Optimierungen dafür sorgen, dass die Entwicklung den Bedürfnissen der Nutzer folgt. [3] [5]



[7]

Abbildung 2: Spring - Blender Open Movie, eines der Projekte des Blender Animation Studio, welches auf YouTube verfügbar ist [7]

2.2 Blender

2.2.1 Programmiersprache und Skriptsprache

Alle Computer besitzen einen oder mehrere CPUs, was die Abkürzung für Central Processing Unit ist und auf Deutsch Prozessor genannt wird. Der CPU kennt nur eine Sprache, und zwar den aus Einsen und Nullen bestehenden Binärcode. Programmiersprachen vereinfachen die Kommunikation zwischen dem Computer und dem Menschen und machen sie verständlicher, denn mittels eines Compilers übersetzt der Computer die Programmiersprache in seine eigene Sprache, den Binärcode. Die Befehle der Programmiersprache werden üblicherweise zeilenweise untereinander programmiert und ergeben gesamthaft den Quellcode. [5] [9]

Die Skriptsprache ist eine Programmiersprache, wobei der Code zur Laufzeit von einem anderen Programm interpretiert wird. [10] [11]

Blender 2.8 verwendet Python für Scripting und API, wobei es mit C, C++ und Python erstellt wurde. API ist die Abkürzung für Application Programming Interface, es ermöglicht, die eigenen Änderungen hinzuzufügen und auf die Programmdaten zuzugreifen. C und C++ werden für die Kernprogrammierung verwendet, der darin geschriebene Code kann nicht verändert werden, ohne Blender neu zu kompilieren. [3] [4] [12]

C ist die Kernsprache, und Python ist die Oberflächensprache. Blender wurde mit der Programmiersprache C entwickelt und ist durch eine Python API ansprechbar. Blender ist mit einem integrierten Python-Interpreter ausgestattet, welcher aktiv bleibt, während Blender läuft. Dieser Interpreter führt Skripte aus, um die Benutzeroberfläche zu erstellen, und wird auch für interne Tools von Blender verwendet. Blender stellt dem Interpreter eigene Python-Module zur Verfügung, diese werden in Skripte importiert, um anschliessend Zugriff auf die Daten und Funktionen von Blender zu haben. Skripte, welche sich mit Blender-Daten beschäftigen, werden von den Modulen importiert, damit sie überhaupt funktionieren können. [3] [12] [13]

2.2.2 Blender 2.8

Die Beta-Version der Blender-Version 2.8 erschien am 26. Februar 2019. Seit dem 30. Juli 2019 ist die stabile Version von Blender 2.8 verfügbar. Was Blender 2.8 so speziell macht, sind die zahlreichen Neuerungen, um eine bessere Benutzerfreundlichkeit zu gewährleisten: Ein neues User Interface und Optimierungen für die Benutzer erleichtern den Umgang mit dem Programm. Neueinsteigern soll der Umgang durch interaktive Tools erleichtert werden, indem Tools, wie beispielsweise das Licht und Kameras, welche zuvor Tastenkombinationen, sogenannte Hotkeys, benötigten, nun einfacher zu nutzen sind. Anfängern wird mit Blender 2.8 der Einstieg in die Welt der 3D-Modellierung erleichtert, indem sie durch einen Toolbar mit einer Symbolleiste und bei der Navigation im Raum durch ein Viewport Gizmo unterstützt werden. [14] [15]



Abbildung 3: Neue kontextbezogene Toolbars [14]

Neu gibt es zudem „Eevee“, was die Abkürzung für Extra Easy Virtual Environment Engine ist, eine sogenannte Real-Time-Render-Engine. Für das Rendern, also die Berechnung eines Bildes der 3D-Objekte, werden durch Eevee völlig neue Möglichkeiten eröffnet. Sie wurde mit Hilfe von OpenGL entwickelt. Die Echtzeit-Render-Engine ist in die funktionsfähige 3D-Software integriert. Ziel ist, eine Echtzeitvorschau von Materialien und Effekten zu erhalten. In Cycles, welche eine Standard-Rendering-Engine von Blender ist, nimmt das Rendern deutlich mehr Zeit in Anspruch, und diese ist somit für eine schnelle Vorschau nicht geeignet. Cycles simuliert die Streuung des Lichts auf eine realistische Art und Weise, Eevee

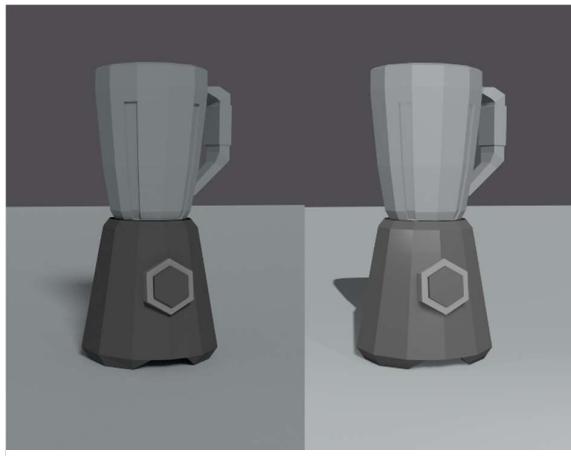


Abbildung 4: Vergleich von Cycles (links) und Eevee (rechts) (eigene Darstellung)

hingegen täuscht die Funktionsweise des Lichts vor und erzeugt so Bilder. Durch die sogenannte Rasterization werden in Eevee die Objekte schattenfrei dargestellt und Schattierungen basierend auf der Position des Lichts überlagert, was allerdings eine Ungenauigkeit der Schatten zur Folge hat. [14] [16] [17] [18] [19]

2.2.3 User Interface von Blender 2.8

User Interface und Hotkeys

Das User Interface, kurz UI, umfasst viele kleine Icons, welche unterschiedliche Funktionen auslösen, doch wenn der Blick ausschliesslich auf das UI gerichtet wird, so wird nur ein kleiner Teil aller Tools in Blender betrachtet. Viele kleinere Optionen liegen verschachtelt in Unterkategorien der einzelnen Modi und sind versteckt hinter weiteren Buttons im Menü. Dies schreckt neue Benutzer oft ab, besonders wenn sie lernen, dass sie ohne die Verwendung von Tastenkombinationen, sogenannten Hotkeys, nur sehr langsam mit ihrem Projekt vorankommen. Blender war ursprünglich für den kommerziellen Gebrauch gedacht, entwickelt für 3D-Artists mit engen Abgabeterminen, die schnell arbeiten müssen. Sind die Hotkeys einmal auswendig gelernt, kann zwischen den Modi, Tools und Menüs viel schneller gewechselt werden. Das UI von Blender sowie die Hotkeys sind individuell anpassbar. [4] [5] Im Folgenden werden die wesentlichsten Bereiche des User Interface erläutert. Da Blender allerdings noch viel mehr Funktionen umfasst, werden nur für diese Arbeit relevante Elemente erwähnt.

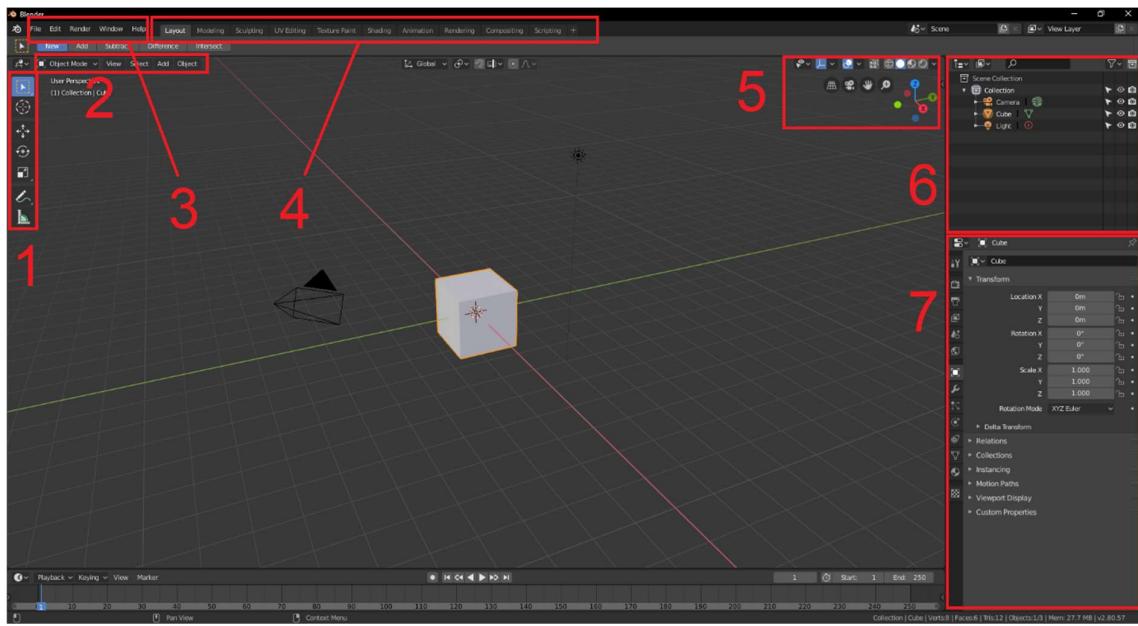


Abbildung 5: User Interface (eigene Darstellung)

In der Abbildung 5 befinden sich, nummeriert mit der Nummer 3, das File-, Edit-, Render-, Window- und Help-Menü. Diese bieten sehr viele Optionen, es können zum Beispiel mit Hilfe des File-Menüs Dateien geöffnet werden, und gerendert wird mit Hilfe des Render-Menüs.

3D-Viewport

Der View zeigt den dreidimensionalen Raum an, in dem sich der Würfel, eine Kamera und eine Lichtquelle in der Abbildung 5 befinden. Im 3D-Viewport befinden sich an den Rändern Menüs, um Tools zur Bearbeitung des Mesh auszuwählen oder die Ansicht zu verändern. Wird ein anderer Tab im Bereich der Nummer 4 gewählt, so ändern sich die Tools ebenso. Um Tools ohne den Gebrauch von Tastenkombinationen zu nutzen, ist ein Toolbar entlang des linken Randes des Viewports zu finden, in der Abbildung 5 nummeriert mit 1. Dieser passt die wählbaren Tools dem aktuellen Modus an.

Manipulation des Mesh

Blender stellt diverse Tools zur Verfügung, um ein Mesh oder je nach Tool auch sämtliche Objekte zu bearbeiten. Durch Scale können die angewählten Bereiche skaliert werden. Extrude sorgt dafür, dass ausgewählte Punkte, Kanten oder Seiten extrudiert werden. Move bewegt die markierten Elemente. Subdivide unterteilt die gewählte Kante oder Fläche. Loop Cut unterteilt das Objekt in mehr Flächen, indem es ringförmig um das Objekt schneidet. Das Knife Tool kann das Mesh beliebig schneiden und lässt so neue Flächen aus bereits bestehenden entstehen.

Modes

Blender enthält diverse Modi, welche unterschiedlichen Zwecken dienen. Der Wechsel zwischen diesen Modi ist ein grundlegender Bestandteil bei der Benutzung von Blender. Der Modus ist in der Abbildung 5 im Feld Nummer 2 zu finden. Wird ein neues Projekt geöffnet, befindet sich Blender automatisch im Object Mode, der das Auswählen von Objekten erlaubt. Ein Wechsel in den Edit Mode ist nötig, damit die einzelnen Punkte, Kanten oder Seiten bearbeitet werden können. Modi können teilweise nur auf bestimmte Objekte bezogen aufrufen werden. Für diese Arbeit relevante Modi sind unter anderem der Weight Paint und der Pose Mode.

Mit Hilfe der Optionen rund um das Viewport Gizmo können Objekte in unterschiedlichen



Abbildung 6: Viewport Gizmo und Ansichtsoptionen (eigene Darstellung)

Ansichtsmodi betrachtet werden, die den Nutzer beispielsweise via Wireframe-Ansicht durch das Mesh blicken lassen oder eine Render-Ansicht anzeigen. Blender verwendet standardmäßig ein rechtshändiges Koordinatensystem. Dabei zeigt in der Frontansicht die x-Achse nach rechts, die y-

Achse nach hinten und die z-Achse nach oben. Die Ansicht ist in der Abbildung 6 **Fehler!** **Verweisquelle konnte nicht gefunden werden.** zu sehen und im rechten oberen Bereich des Viewports zu finden. Darunter befinden sich weitere Tools, um die Ansicht im 3D-Viewport bewegen und drehen zu können oder die Kameraansicht zu erhalten.

Szene und Properties

Im sechsten Fenster in der Abbildung 5 sind die sich im 3D-Viewport befindenden Objekte aufgelistet und können umbenannt, gruppiert oder unsichtbar gemacht werden. Darunter, im siebten Bereich des Screenshots in Abbildung 5, befindet sich das Properties-Panel mit jeglichen einstellbaren Optionen, welche beispielsweise die Eigenschaften des Materials, die Modifiers, die Bones oder die Szene betreffen.

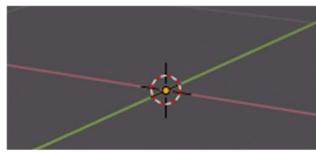


Abbildung 7: 3D-Cursor und
Pivot Point am Ursprung
(eigene Darstellung)

Neue Objekte werden an der Position des 3D-Cursors angelegt und verwenden als Bezugspunkt für Manipulationen einen kleinen orangefarbenen Punkt, den Pivot Point, welcher bei einem neu erstellten Mesh an der Stelle des 3D-Cursors erscheint. [4] [5]

Mirror-Modifier

Der Mirror-Modifier ist eines der wichtigsten Elemente beim Modellieren eines Charakters, da dieser in der Regel eine Symmetrieachse besitzt. Mit Hilfe dieses Modifiers muss nur eine Seite modelliert werden, die andere Hälfte wird durch die Spiegelung an der Spiegelachse, die durch den Pivot Point geht, automatisch konstruiert. Zu finden ist der Modifier im Modifiers Panel auf der rechten Seite [5].



Abbildung 8: Mirror-Modifier
(eigene Darstellung)

Rigging und Weight Painting

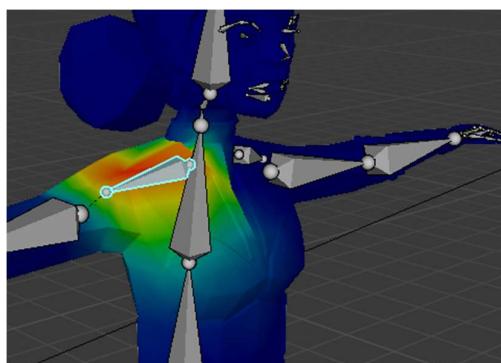


Abbildung 9: Weight Painting (eigene Darstellung)

Um ein 3D-Modell bewegen zu können, braucht es ein Rig, ein aus Bones bestehendes zusammenhängendes Gebilde. Der Prozess zur Erstellung eines Rig wird Rigging genannt. Bones besitzen andere Modes als Meshes. Im Pose Mode kann das Rig posiert werden. Zwischen dem Rig und dem Mesh muss eine Beziehung hergestellt werden, wobei das Rig der Parent ist und das Mesh das Child, dieses lässt sich vom

Parent, also vom Rig, beeinflussen. Mit Hilfe des Weight Paint Mode kann das Mesh bemalt werden, um die Einflussbereiche eines Bone zu bestimmen. Wird ein Bereich beeinflusst, wird dies durch eine rote Stelle sichtbar. Ist dieser blau, so wird er nicht beeinflusst. Alles Dazwischenliegende besitzt eine andere Farbe und lässt sich dehnen und biegen. [4] [5]

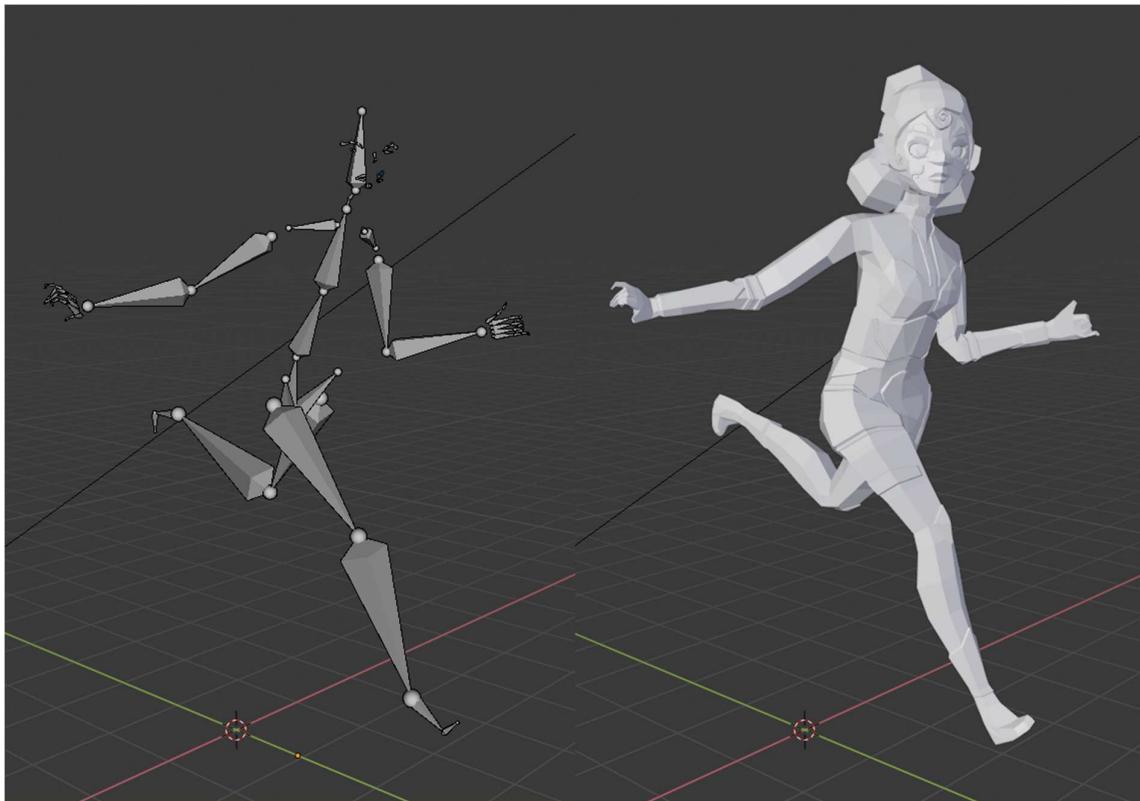


Abbildung 10: Pose des Rig und des gesamten Modells nach Parenting und Weight Painting (eigene Darstellung)

Materials

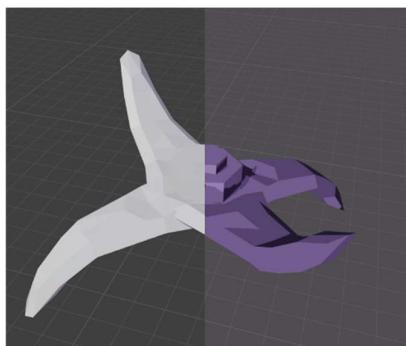


Abbildung 11: Objekt, links ohne und rechts mit eigenem Material (eigene Darstellung)

Ebenfalls im Properties-Panel zu finden, sind Materials. Materials bestimmen die Oberfläche eines 3D-Objekts und können unterschiedliche Eigenschaften aufweisen, was sie zum Beispiel rau oder glatt aussehen lässt. [5] Blender stellt unterschiedliche Methoden zur Verfügung, um einem Objekt ein Material zu geben, und liefert gewisse Materials bereits mit dem Download mit.

Um einen Shader für die Materials erstellen zu können, muss zum Shading-Tab gewechselt werden.

2.3 Shader

2.3.1 Funktionsweise eines Shaders

Ein Shader ist ein Programm, welches für die Berechnung von Licht- und Schattierungseffekten verwendet wird, und kann in einer bestimmten Shading-Sprache geschrieben werden. In OpenGL zum Beispiel heisst diese OpenGL Shading Language, was eine C-ähnliche Sprache ist. Blender benutzt ein Node-basiertes System zur Definition von Shadern, welches im Unterkapitel zu Nodes in Blender genauer erklärt wird. Grundsätzlich läuft der Shader auf einer GPU. Er ist ein Teil der OpenGL-Rendering-Pipeline, welche aus einer 3D-Szene ein Bild berechnet. Dieses kann danach auf dem Bildschirm angezeigt werden. Im Gegensatz zur Render-Engine Cycles, welche ein Raytracer ist, der CUDA und OpenCL verwendet, benutzt Blender 2.8 neu auch noch die neue Render-Engine Eevee basierend auf OpenGL für Real-Time-3D-Darstellungen. [20] [21]

Der Vertex-Shader und der Fragment-Shader sind für OpenGL elementare Shader, welche in dieser Arbeit genauer erklärt werden. Perspective Division und Backface Culling werden in dieser Arbeit nicht behandelt. [21] [22] [23]



Abbildung 12: Rendering-Stages (eigene Darstellung)

Der Vertex-Shader

Vertex-Shader werden eingesetzt, um die Position von Vertex-, Farb- und Texturkoordinaten im Rendering-Prozess zu modifizieren. Jeder Vertex kann mit vielen unterschiedlichen Variablen definiert werden, ein Vertex kann beispielsweise durch seine Position im Koordinatensystem unter Verwendung der x-, y-, z-Koordinaten definiert werden. Vertex-Shader können nur Änderungen an bereits vorhandenen Vertices vornehmen, aber keine neuen erstellen, da dieser Shader pro Vertex aufgerufen wird. [5] [24] [25] [26]

Dieser Shader dient dazu, die Geometrie der 3D-Objekte zu beeinflussen, dadurch werden die Koordinaten der Oberflächen transformiert. Das bedeutet, dass eine Punktmenge verschoben, gedreht oder skaliert wird. Somit kann man die Form und die Beleuchtung von Objekten beeinflussen. Im Gegensatz zu OpenGL weiss Blender, wie das Licht definiert ist, welches der Vertex-Shader ausführen kann. Der Vertex-Shader ist da, um die Positionen der Vertices des relativen Objekts aus dem Koordinatensystem, welches alle 3D-Objekte besitzen, ins globale Koordinatensystem umzuwandeln. Das globale Koordinatensystem in

einem Programm ist ein dreidimensionales, orthogonales und rechtshändiges Koordinatensystem, welches die Achsen x, y und z besitzt. Nach der Umwandlung in das globale Koordinatensystem können sie in das Kamerakoordinatensystem umgerechnet werden. [24] [25] [26] [27]

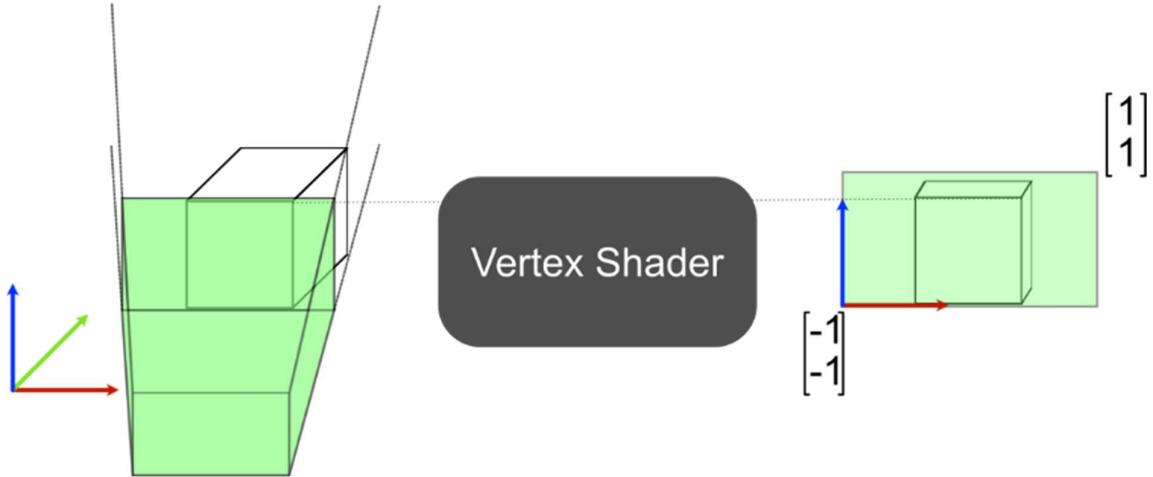


Abbildung 13: Ein Vertex-Shader ([23]; eigene Darstellung)

Der Fragment-Shader

Der Fragment-Shader, welcher auch Pixel-Shader genannt wird, dient der Veränderung der zu rendernden Fragmente. Ein Fragment besteht aus einer Position im Koordinatensystem, mit x, y-Ansichtsfenster und in der Tiefe z, und allen interpolierten Daten aus den vorherigen Phasen [28]. So kann mit einem solchen Shader zum Beispiel eine realistische Darstellung der Oberflächen- und Materialeigenschaften erreicht werden [5]. Die Pixel des endgültigen Bildes, welches angezeigt wird, sind möglicherweise aus mehreren Fragmenten zusammengesetzt. Dies ist beispielsweise der Fall, wenn mehrere Objekte gleichzeitig gesehen werden können, weil diejenigen im Vordergrund transparent sind, denn die Shader verarbeiten die Objekte unabhängig voneinander. [29]

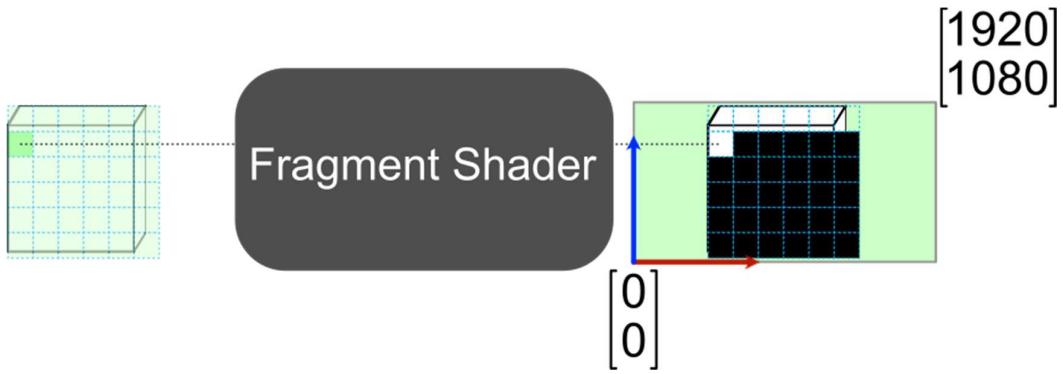


Abbildung 14: Ein Fragment-Shader ([23]; eigene Darstellung)

2.3.2 Phong Reflection Model

Mit diesem Model wird für jeden Punkt auf der Oberfläche des 3D-Objekts die Beleuchtungsstärke berechnet. Das Phong Reflection Model, nach seinem Entwickler Büi Tường Phong benannt, wird am häufigsten in der Computergrafik verwendet. Die modellierte Beleuchtung besteht aus drei Teilen: Ambient, Specular und Diffuse. [30] [31] Die letzteren beiden werden in dieser Arbeit genauer erläutert.

Diffuse Lighting

Diffuse Lighting erzeugt eine höhere Helligkeit des Objekts, je näher die Fragmente an den Lichtstrahlen einer Lichtquelle ausgerichtet sind. [32]

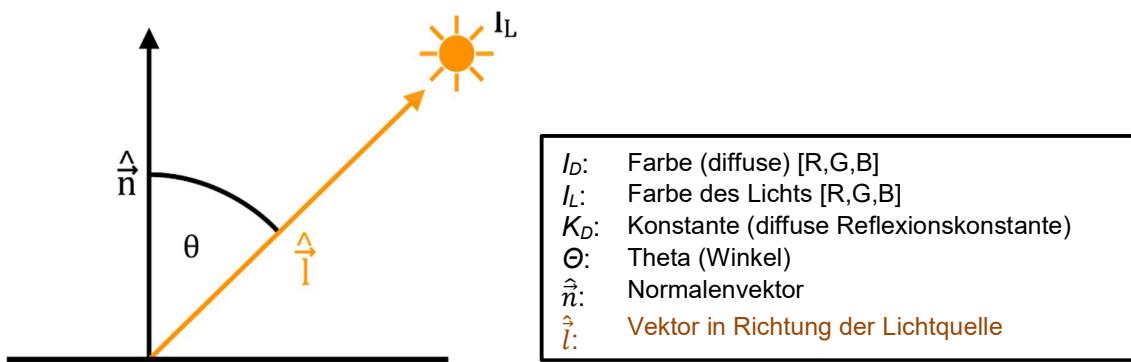


Abbildung 15: Berechnung des Diffuse Lighting ([32]; eigene Darstellung)

$$I_D = I_L \cdot K_D \cdot \cos(\theta) \quad (2.1)$$

In der Abbildung 15 befindet sich eine Lichtquelle mit einem Vektor in Richtung der Lichtquelle, der von einem bestimmten Fragment des 3D-Objekts aus zur Lichtquelle gerichtet ist. Um den Winkel θ zwischen dem Lichtstrahl und dem Fragment zu berechnen, wird der Normalenvektor der Fläche verwendet, welcher senkrecht zur Oberfläche steht. Die Farbe des Diffuse Lighting wird anhand der Formel (2.1) berechnet. [33]

Mit Hilfe der Zwischenwinkelformel und der Einheitsvektoren wird der Cosinus des Zwischenwinkels θ berechnet:

$$\cos(\theta) = \frac{\vec{n} \odot \vec{l}}{|\vec{n}| \cdot |\vec{l}|} \quad (2.2)$$

Der Winkel θ zwischen dem Normalenvektor und dem Vektor in Richtung der Lichtquelle wird mit dem Skalarprodukt berechnet. Da mit Einheitsvektoren gerechnet wird und diese somit eine Länge von 1 aufweisen, fällt der Nenner des Bruchs weg:

$$\cos(\theta) = \vec{n} \odot \vec{l} \quad (2.3)$$

Somit kann die Formel für das Diffuse Lighting (2.1) folgendermassen umgeformt werden:

$$I_D = I_L \cdot K_D \cdot \vec{n} \odot \vec{l} \quad (2.4)$$

Das Diffuse Lighting (I_D), in der Formel (2.4), ergibt sich aus der Farbe des Lichts (I_L) multipliziert mit der diffusen Reflexionskonstante und dem Skalarprodukt der beiden Einheitsvektoren \hat{n} und \hat{l} . [32]

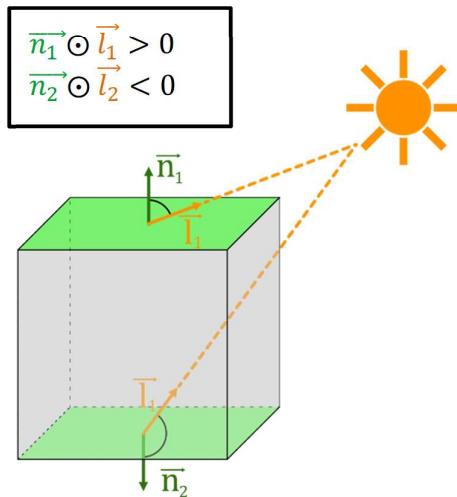


Abbildung 16: Beispiel von positivem und negativem Skalarprodukt ([34]; eigene Darstellung)

Licht reflektiert wird. [32]

Die Farbe des Lichts besteht aus den drei Farbkanälen, Rot, Grün und Blau. Die Konstante K_D liegt zwischen 0 und 1 und beschreibt, wieviel Licht reflektiert wird. Der übrige Faktor, also $1 - K_D$, wird absorbiert. [33]

Liegen die Vektoren aufeinander, so beträgt der Zwischenwinkel 0° , der Cosinus von 0° ergibt 1. Wenn der Lichtstrahl senkrecht zur 3D-Objektoberfläche steht, besitzt das Licht die grösste Ausstrahlungswirkung. Stehen die beiden Vektoren rechtwinklig zueinander, ergibt der Cosinus dieses Winkels 0. Je näher der Wert des Cosinus bei 1 liegt, desto mehr Licht und desto heller wird der Punkt. Je grösser der Winkel ist, desto geringer ist der Effekt, den das Licht auf die Farbe des Fragments haben wird. Wenn das Skalarprodukt negativ ist, wird 0 angenommen, da „negatives“ Licht nicht erwünscht ist und durch eine Null kein

Specular Reflection

Mit der Specular Reflection werden heller beleuchtete Stellen auf dem Objekt, die Glanzpunkte, dargestellt. Diese Reflexionen sind, wie das Diffuse Lighting, abhängig von der Richtung des Lichtstrahls und des Normalenvektors des Objekts, zusätzlich jedoch noch vom Blickwinkel, in welchem auf die Oberfläche geschaut wird, beziehungsweise von der Position der Kamera. Vergleichbar mit einem Spiegel sind die Stellen, welche Licht reflektieren, die Stellen auf dem Objekt, die am stärksten beleuchtet sind [32] [34]. Der Reflexionsvektor wird mit der Reflektionsformel XE "Reflektionsformel:Angenommen sei ein Strahl, der aus der Richtung v (normiert) auf eine Oberfläche mit Normale r (auch normiert) trifft. Dann bestimmt sich die normierte Reflexions-Richtung durch die Formel: durch das Reflektieren der Lichtrichtung an der Stelle des Normalenvektors berechnet. Anschliessend wird der Zwischenwinkel des Reflexionsvektors und des Vektors vom Auge zum Berechnungspunkt bestimmt.

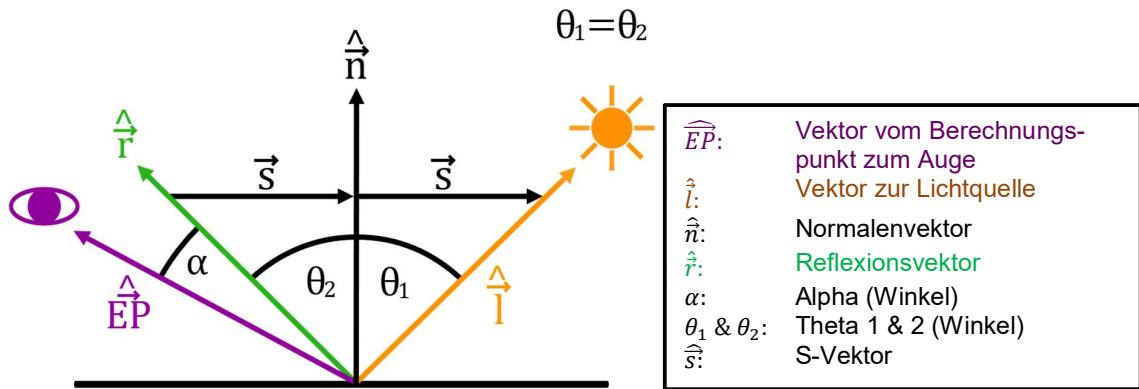


Abbildung 17: Berechnung der Specular Reflection ([32]; eigene Darstellung)

Die Formel zur Berechnung der Specular Reflection lautet wie folgt:

$$I_S = I_L \cdot K_S \cdot \cos^k(\alpha) \quad (2.5)$$

Der Cosinus des Zwischenwinkels wird wie beim Diffuse Lighting mit Hilfe der Zwischenwinkelformel berechnet:

$$\cos(\alpha) = \frac{\overrightarrow{EP} \odot \vec{r}}{|\overrightarrow{EP}| \cdot |\vec{r}|} \quad (2.6)$$

Durch das Rechnen mit Einheitsvektoren fällt der Nenner des Bruchs weg:

$$\cos(\alpha) = \widehat{\overrightarrow{EP}} \odot \widehat{\vec{r}} \quad (2.7)$$

Je grösser der Zwischenwinkel des Vektors von der Kamera zum Berechnungspunkt und des Reflexionsvektors, desto weniger Licht wird von der Kamera erfasst. Liegen die Vektoren aufeinander, so beträgt der Zwischenwinkel 0° , der Cosinus von 0° ergibt 1. Stehen die beiden Vektoren rechtwinklig zueinander, beträgt der Cosinus dieses Winkels 0. Je näher der Wert des Cosinus bei 1 steht, desto mehr Licht wird vom Auge erfasst und desto heller wird der Punkt. [35]

Es kann für den Cosinus des Zwischenwinkels das Skalarprodukt der beiden normalen Vektoren $\widehat{\overrightarrow{EP}}$ und $\widehat{\vec{r}}$ in die Formel (2.5) eingesetzt werden:

$$I_S = I_L \cdot K_S \cdot \left(\widehat{\overrightarrow{EP}} \odot \widehat{\vec{r}} \right)^k \quad (2.8)$$

Das Licht der Specular Reflection lässt sich aus der Farbe des Lichts multipliziert mit der Specular-Reflection-Konstante und dem Skalarprodukt der beiden Einheitsvektoren $\widehat{\overrightarrow{EP}}$ und $\widehat{\vec{r}}$, welches mit k hochgerechnet wird, berechnen. Die Farbe des Lichts besteht, wie beim Diffuse Lighting, aus den drei Farbkanälen. Die Konstante K_S liegt, wie K_D , zwischen 0 und 1 und beschreibt, wie stark der Glanzeffekt ist. Je grösser k , desto grösser sind die Glanzpunkte. [33]

Werden beide Reflexionsmodelle gleichzeitig verwendet, so erhält man eine Oberfläche mit einem Farbverlauf sowie auch die Glanzpunkte.

2.3.3 Cel Shading

Verwendung des Cel Shader



Abbildung 18: The Legend of Zelda: The Wind Waker, 2002 erschienen [36]

Cel Shading, auch Toon Shading genannt, ist ein Computer-Rendering-Stil, welcher den Schattengradienten des Renderings durch einheitliche Farben und Schatten ersetzt. Dadurch erhalten die 3D-Modelle einen Look, als wären sie im Stil eines Comics handgezeichnet. Diese 3D-Modelle wirken umso Comic-hafter, wenn sie zusätzlich noch schwarze, dicke Außenkanten erhalten.

Auf Texturen wird oft verzichtet, somit werden die weichen Übergänge zwischen Licht und Schatten vermieden und nur wenige Abstufungen zwischen Licht und Schatten gemacht. Der englische Begriff Cel ist eine Kurzform für „Celluloid“, also die Kunststoffverbindung Zelluloid, und Toon wird aus dem englischen „Cartoon“ abgeleitet. [4] [5] [37] [38] [39]

Für die Schatten der 3D-Modelle werden in der Regel drei bis vier Helligkeitsstufen verwendet – weiss, hellgrau und dunkelgrau –, und auf der Oberfläche wird meistens auf Texturen verzichtet und es werden nur einzelne Farbstufen verwendet. Die Grauwerte werden über den Winkel berechnet, welcher zwischen der Normalen des Polygons und dem Vektor liegt, welcher von einem Fragment des Polygons zur Lichtquelle geht. Der Cosinus des Winkels, welcher durch das Skalarprodukt der beiden Vektoren berechnet wird, bestimmt in welcher Helligkeitsstufe sich das Fragment befinden soll. [34] [37]

Unterschied zwischen Phong Shader und Cel Shader

Wie beim Phong Shader wird das Diffuse Lighting eines Fragments auch beim Toon Shader über das Phong-Beleuchtungsmodell berechnet. Jedoch liegt die ausschlaggebende Besonderheit des Toon Shading im Vergleich zum Phong Shading darin, dass es anstelle eines glatten Farbverlaufs eine bestimmte Anzahl farblicher Abstufungen gibt. Wenn beim Toon Shader das Glanzlicht einen bestimmten Wert übersteigt, wird es auf Weiss gestellt, somit besitzt das Glanzlicht eine Kontur. Im Gegensatz zum Cel Shading ist der Phong Shader ein realistisches Shading-Verfahren. [4] [40] [41]

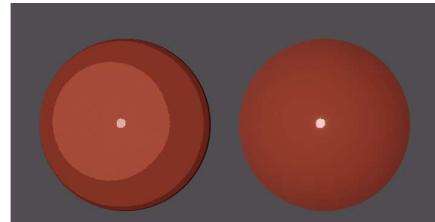


Abbildung 19: Cel Shader (links) und Phong Shader (rechts) (eigene Darstellung)

2.3.4 Nodes in Blender

Nodes sind im Wesentlichen eine einfachere Art der Shader-Programmierung, wobei die einzelnen Nodes Boxen mit In- und Outputs sind. Nodes sind Funktionseinheiten, die durch den Node Editor, welcher Teil der Benutzeroberfläche von Blender ist, miteinander verknüpft werden können. Der Input-Node ist der Startpunkt, welcher beispielsweise eine Farbe

enthaltet. Weitere Nodes werden an den Input-Node angehängt. Diese weiteren Nodes können die Farbe modifizieren oder einzelne Farbkanäle verändern. Durch Verknüpfen der Nodes können Veränderungen erzielt werden, die die Farbe oder Schattierung des Objekts beeinflussen. [42]



Abbildung 20: Verbundene Nodes (eigene Darstellung)

Node Groups

Eine Node Group fasst die gewählten Nodes zu einem Node zusammen. Nodes zu gruppieren, kann bei der Übersichtlichkeit des erstellten Shaders helfen, und die Node Groups können kopiert, auf andere Materialien übertragen und so erneut verwendet werden. [43]

Geometry Node

Der Geometry Node ist dafür zuständig, Informationen zur Geometrie eines Punktes zu geben. Alle Koordinaten von Vektoren sind im World Space vorhanden. Wichtig für den erstellten Shader ist der Normalen-Output, welcher die Normale auf der Oberfläche des Objekts angibt. [44]

Specular BSDF Node

Der Specular BSDF Node fasst mehrere Berechnungen zu einem einfach zu bedienenden Node zusammen. Der Benutzer muss keine Formeln eingeben, denn der Node muss nur noch mit anderen Nodes verknüpft werden. Vom Geometry Node erhält dieser Node Informationen zur Geometrie von Punkten. Im Node sind einige Parameter verstellbar, wie die Roughness, welche angibt, wie rau die Oberfläche des Objekts ist, was eine Rolle spielt für die Menge des reflektierten Lichts. [45]



Abbildung 21 Specular BSDF Node (eigene Darstellung)

Shader To RGB

Der Shader To RGB Node wird normalerweise für nicht-fotorealistisches Rendering verwendet, um zusätzliche Effekte zum Output des BSDF Node hinzuzufügen. So kann beispielsweise ein ColorRamp Node mit diesem Node verbunden werden, um einen flexiblen Cel Shader zu erstellen. [46]

ColorRamp Node

Der ColorRamp Node wird dazu verwendet, Werte zu Farben unter Verwendung eines Farbverlaufs zuzuordnen. Dieser kann vom Benutzer eingestellt werden, dabei lassen sich neue Ramps hinzufügen und so mehr Abstufungen hervorbringen. Es werden Grenzen festgelegt, die Farbwerte bis zu diesem Wert werden dunkler dargestellt, alle über dem Grenzwert liegenden Werte werden als die hellere Farbe ausgegeben. Diese Grenzen spielen bei einem Cel Shader eine wesentliche Rolle. Der Output des Shader To RGB Node ist somit eine Oberflächenfarbe, welche sich aus den BSDFs und der Beleuchtung berechnen lässt. [47]

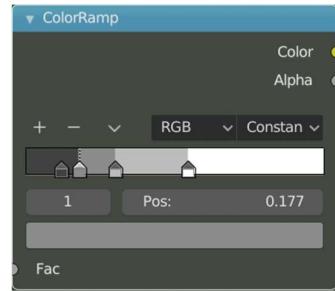


Abbildung 22: ColorRamp Node
(eigene Darstellung)

Multiply Node

Der Multiply Node multipliziert zwei Farben. Blender verwendet Werte zwischen 0.0 und 1.0 für die Farben. Dieser Vorgang muss nicht normalisiert werden, da die Multiplikation von zwei Termen zwischen 0.0 und 1.0 immer ein Ergebnis zwischen 0.0 und 1.0 ergibt. [48]

Mix Node (Add)

Der Mix Node ist für das Zusammenfügen von Farben zuständig. Es gibt mehrere Methoden, die Farben zu mixen, was zu unterschiedlichen Resultaten führen kann. Add addiert die Farbkanäle einzeln. Das Hinzufügen von Blau zu Blau belässt es blau, aber wird Blau zu Rot addiert, entsteht ein Violett. Weiss besitzt bereits den maximalen Wert einer Farbe, wird also Blau hinzuaddiert, so bleibt es weiss. [49]

Diffuse BSDF Node



Abbildung 23: Diffuse BSDF Node
(eigene Darstellung)

Der Diffuse BSDF Node wird verwendet, um Lambertian oder Oren-Nayar Diffuse Reflection hinzuzufügen. Welche Art der Reflexion dabei zum Einsatz kommt, hängt von der Roughness ab. Liegt der Wert der Roughness bei 0.0, so handelt es sich um Lambertian Diffuse Reflection. Alle höheren Werte aktivieren Oren-Nayar Diffuse Reflection, auf welche in dieser Arbeit jedoch nicht weiter eingegangen wird, da diese Art der Reflexion komplexer ist. [50]

RGB Node

Der RGB Node besitzt keine Input-Verbindungspunkte, er dient lediglich der Farbwahl, in dem er das Color Picker Widget zur Verfügung stellt. Der Output ist somit ein RGB-A-Farbwert. [51]

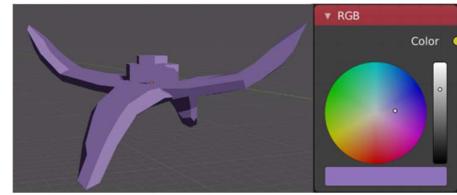


Abbildung 24: RGB Node und Objektfarbe
(eigene Darstellung)

Emission Node

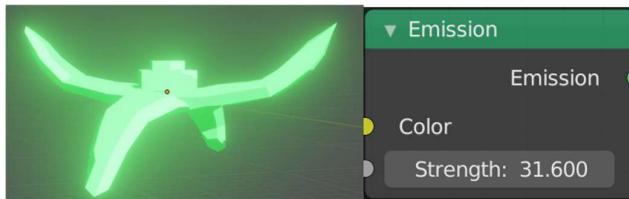


Abbildung 25: Leuchteffekt und Emission Node (eigene Darstellung)

Der Emission Node wird verwendet, um den Lambertian Emission Shader zu ergänzen. Dieser kann beispielsweise für Material- und Lichtflächenausgaben verwendet werden. Die Input-Farbe ist jene, die als Farbe des emittierten

Lichts ausgegeben wird. Die Stärke dieses Lichts ist verstellbar. Bei Materialien stellt ein Wert von 1,0 sicher, dass das Objekt im Bild genau die gleiche Farbe wie die Farbeingabe hat, das heisst, dass es die Oberfläche schattenfrei darstellt und somit leuchtet. [52]

Material Output Node

Der Material Output Node dient zur Ausgabe von Informationen zum Oberflächenmaterial an ein Oberflächenobjekt. Der Surface-Input ist für die Schattierung der Oberfläche des Objekts zuständig. [53]

2.4 Low Poly

Poly ist die Abkürzung für Polygon beziehungsweise Vieleck, was eine geometrische Form ist, welche lediglich aus durch Kanten verbundenen Punkten besteht. Ein Polygonnetz entsteht durch Verknüpfung von mehreren Polygonen und dient als Grundlage eines 3D-Objekts im Bereich der 3D-Computergrafik. Bei Verwendung des Low-Poly-Stils weist das Polygonnetz eine geringere Dichte von Polygonen auf. Ein Charakter, welcher Low Poly ist, hat meist Polygone, die gross genug sind, um sie erkennen zu können. 3D-Modelle können auch aus einer hohen Anzahl Polygone bestehen, um beispielsweise ein realitätsnahe Gesicht zu modellieren, hierzu werden über 6'000 Polygone benötigt. Diese Methode, welche für realistische 3D-Modelle eingesetzt wird, nennt man High Poly. Diese

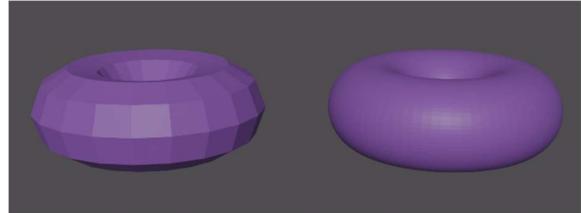


Abbildung 26: Low und High Poly (eigene Darstellung)

Begriffe sind allerdings relativ, denn was heute als Low Poly gilt, war vor einigen Jahren noch High Poly. [4] [54]

Mit Low Poly beschäftigen sich vor allem die Entwickler von Games und Game-Designer, denn diese geometrischen Flächen benötigen sie als Grundlage zur Erstellung von 3D-Gegenständen, welche in Echtzeit berechnet werden. [55]



Abbildung 27: Super Mario 64, ein Videospiel, welches 1996 erschien, gilt heutzutage als Low Poly [56]

Die 3D-Computerspiele dienten als Grundlage für die Entstehung von Low Poly, welche schon früh in den 1990er Jahren entstand. Bei den Computern dieser Zeit waren die Rechenleistungen begrenzt, somit konnten nur vereinfachte 3D-Modelle erstellt werden mit so wenigen Polygone wie möglich. [54]

Heute ist Low Poly nicht nur aufgrund seines eigenständigen visuellen Stils zu

einem Trend geworden, Low-Poly-Darstellungen sind zudem auch simpler und somit zeit- und ressourcensparend, deshalb ist es nicht verwunderlich, dass sie immer mehr Anwendung in der Videospiele-Branche finden. Aufgrund der verminderten Anzahl an Polygone in einem Low-Poly-Mesh ist die benötigte Rechenleistung geringer. [54]

Ein Mesh besteht aus Punkten, den Vertices, welche jeweils aus drei Fließkommazahlen bestehen, was $3 * 32$ Bits sind. Da Meshes einen kleinen Anteil der Dateigröße ausmachen, spielt die Anzahl der Polygone keine grosse Rolle in der Dateigrößenoptimierung, wobei die Texturen hierbei wichtiger sind. Für Berechnungen ist die Anzahl der Vertices allerdings sehr



Abbildung 28: SUPERHOT VR, ein Videospiel, welches 2016 erschienen ist [57]

relevant, da von dieser die benötigte Rechenleistung des PCs abhängt und bei grösserer Anzahl längere Berechnungszeiten benötigt werden. Somit können auch PCs mit einer niedrigeren Rechenleistung die Modelle deutlich schneller berechnen. Auch in Spielen, die sich nicht primär am Low-Poly-Stil orientieren, findet

man 3D-Modelle mit weniger Polygonen: Objekte, welche sich im Hintergrund befinden oder unwichtiger sind, weisen weniger Details auf, um Ressourcen zu sparen. Damit diese Modelle dennoch eine ästhetische Wirkung haben, wird oft mit Licht, Schattierungen und Texturen gearbeitet. [54]

3 Erstellungsprozess des Charakters

3.1 Charakterdesign



Abbildung 29: Entwurf des in Blender zu erstellenden Charakters (eigene Darstellung)

In der Abbildung 29 ist das Konzept des zu erstellenden 3D-Modells ersichtlich. Der Entwurfsprozess war durch diverse Stilrichtungen wie den Steampunk, als auch den Cyberpunk beeinflusst, die Letztere blieb in der Grundidee des finalen Konzepts erhalten: In einer dystopischen Zukunft braucht es Heldinnen und Helden, die die Welt vor einer Apokalypse bewahren.

Die Heldenin soll mutig und feminin wirken. Ihr Erscheinungsbild soll den Betrachter in eine Zukunft versetzen, in welcher die Menschen immer mehr von Maschinen verdrängt werden und ihr Leben durch deren Einfluss durchgehend bestimmt ist. Die beiden Platten, welche Teil des Anzugs der Figur sind und auf ihren Hüften sitzen, erinnern an einen Rock, ein Kleidungsstück, welches als elegant und weiblich empfunden wird, aber auch an einen Schild, um sich vor Gefahr zu schützen. Die violetten Bestandteile und die Augen können zum Leuchten gebracht werden und zeigen mit ihrem ungewöhnlichen und ominösen Effekt, dass diese Heldenin kein gewöhnlicher Mensch ist, sondern eine roboterartige Lebensform, die nur noch durch ihr menschliches Gesicht an einen Menschen erinnert. Die Farben sollen das Interesse wecken und sind daher kontrastreich gewählt, wobei es sich bei der Darstellung nur um einen Entwurf und nicht um eine finale Abbildung der Figur handelt. Während der Entwicklung des 3D-Modells werden sich aus ästhetischen Gründen gewisse Aspekte im Charakterdesign noch stark verändern.

3.2 Erstellung des 3D-Modells

Blender wurde von der Blender-Webseite unter <https://www.blender.org/download/> heruntergeladen und installiert. Die Installation variiert je nach Betriebssystem leicht, unter https://docs.blender.org/manual/en/latest/getting_started/installing/index.html sind für Linux, Windows und MacOS Hinweise zur Installation zu finden.

Im Folgenden werden die wichtigsten Aspekte des Prozesses beschrieben, jedoch wird nicht weiter erwähnt, dass dieser vielfach unterbrochen wurde durch kleinste Aktionen wie:

- das Auswählen eines Objekts
- einzelne Verschiebungen und Erstellen von Punkten
- Wechsel zwischen den Modi.

Nach Aufruf von Blender erschien das Startobjekt, ein Würfel. Dieser wurde entfernt. Bevor die Vorlage aus dem Ordner in das geöffnete Blender-Fenster hinübergezogen wurde, fand ein Wechsel in die Frontansicht statt, sodass die Kamera auf die yz-Ebene blickte.

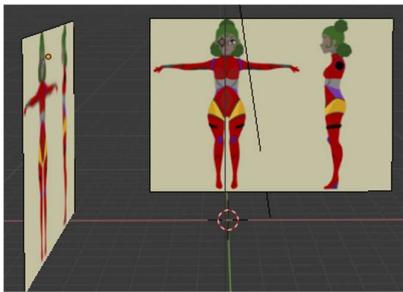


Abbildung 30: Eingefügte Vorlagen im 3D-Viewport (eigene Darstellung)

Der Einfügeprozess der Vorlage wurde in Bezug auf die xz-Ebene wiederholt, um eine Seitenansicht des Bildes zu erhalten. Da eine Vorlage verwendet wurde, musste darauf geachtet werden, dass die Spiegelachse des Körpers auf der z-Achse liegt, denn die Fläche befand sich durch den Wechsel in die Frontansicht vor dem Einfügen bereits in der yz-Ebene. Dieses Detail spielt bei der Benutzung eines auf Symmetrie basierenden

Modifiers, welcher sich am Pivot Point orientiert, eine wichtige Rolle [58]. Damit die Werte der Spiegelachse oder die sonstigen Koordinaten nicht verändert werden mussten, wurde besonders zu Beginn des Prozesses streng auf die Einhaltung der obigen Prozesse geachtet.

Erstellung des Mesh

Die Basis des Charakters lag in einer simplen Zylinderform. Nach dem Hinzufügen des Mesh wurde mit Hilfe der Option „Adjust last operation...“ die Anzahl der Vertices auf 8 reduziert, was beim Zylinder zu einer verminderten Anzahl an Ecken an der Grund- und Deckfläche führte, dies liess den Zylinder eckiger wirken.

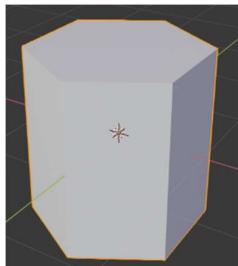


Abbildung 31: Zylinder
(eigene Darstellung)

Wurde der Pivot Point durch das Verschieben des Objekts im Object Mode verschoben, so musste dieser wieder auf den Ursprung zurückgesetzt werden. Der Mirror-Modifier wurde ausgewählt und der Zylinder wurde „geklont“. So konnte die Basis der Beine entstehen. Die beiden Zylinder wurden skaliert, sodass sie nur noch so breit wie die Beine waren. Im Edit Mode wurde das Mesh bearbeitet. Die Grund- und

Deckfläche des Zylinders wurden gelöscht, was einen hohlen Zylinder zum Vorschein brachte. Es wurde die Gitteransicht gewählt, damit alle Vertices ausgewählt werden konnten. Durch

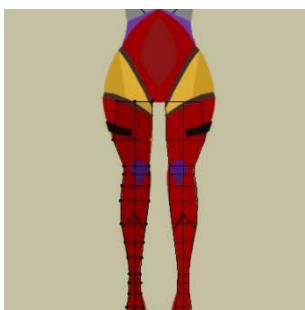


Abbildung 33: Erstellung der
Beine (eigene Darstellung)

das Klicken der linken Maustaste und Ziehen wurde eine Auswahlbox hervorgerufen. So konnten alle sich am oberen Rand befindenden Vertices ausgewählt werden. Durch Extrudieren konnte das Mesh nach oben erweitert werden, wobei es eine Art Naht an der Startstelle erhielt. Diese „Naht“ bestand aus den Ausgangspunkten für das Extrudieren. Da das Bein des Charakters kurvig ist und nicht aus einer geradlinigen Röhre besteht, wurden die Vertices, die extrudiert wurden, skaliert. So wurde der „Ring“ aus Vertices enger oder weiter und das Mesh glich immer mehr den beiden Beinen der Helden.

Damit die Figur aus jedem Blickwinkel authentisch wirkt, genügte ein Wechsel zwischen der Seiten- und Frontansicht nicht, besonders wenn detaillierte Stellen modelliert wurden. Unter häufigen Perspektivenwechseln und durch Verschiebungen der Vertices, Verwendung von Loop-Cuts und des Knife-Tools wurden die Schuhe des Anzugs geformt. Am oberen Ende der Oberschenkel, im Schrittbereich, mussten die Vertices zusammengeführt werden, sodass sie sich überlappten und nach Anwendung des Mirror-Modifiers automatisch verschmolzen. Durch das Schneiden der Edges und Verschieben der Vertices auf der xy-Ebene wurde die Basis des Torsos geformt.

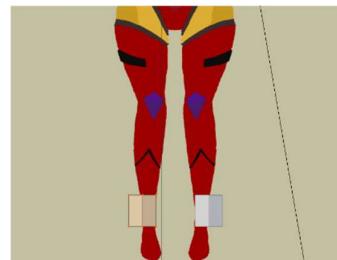


Abbildung 32: Gespiegelte
Zylinder (eigene Darstellung)

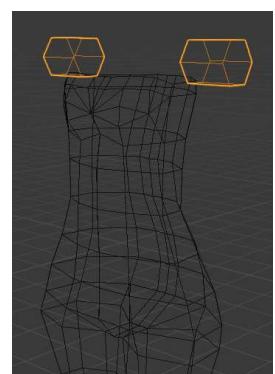


Abbildung 34: Oberkörper
und Arme (eigene
Darstellung)

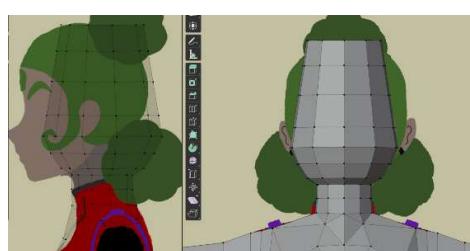


Abbildung 35: Erstellungsprozess des Kopfs
(eigene Darstellung)

Die Vertices am oberen Rand wurden extrudiert. Das Unterteilen von Edges wurde benötigt in Bereichen mit mehr Detail, wie dem Dekolleté- und Schulterbereich. Die Arme wurden ähnlich wie die Beine aus Zylindern modelliert. Die Vertices an den

Schnittstellen mit dem Körper wurden miteinander verschmolzen. Die Hände haben viel mehr Feinheiten, welche trotz des Low-Poly-Stils nicht verloren gehen durften. Ausgangspunkt der Hände waren die Enden der Arme, zylinderförmige Meshes, die extrudiert wurden bis zu den Fingerspitzen der Vorlage. Unter Verwendung des Knife-Tools wurde eine Rechteckfläche ausgeschnitten und extrudiert. So wurde der erste Finger modelliert: der Daumen. Dieselbe

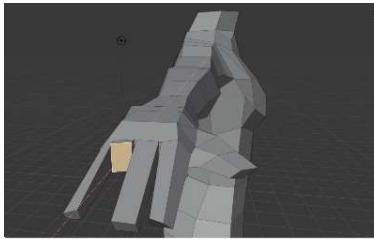


Abbildung 36: Erstellungsprozess der Hände (eigene Darstellung)

Vorgehensweise wurde bei den restlichen Fingern angewandt. Durch Loop-Cuts wurden die Finger in Segmente unterteilt, damit sie, wie richtige Finger, an den Gelenken bewegt werden können. Die Ober- und Unterseite der Hände wurde durch weitere Schnitte im Mesh unterteilt, damit überschüssige Vertices an anderen Stellen der Hand wieder verschmolzen

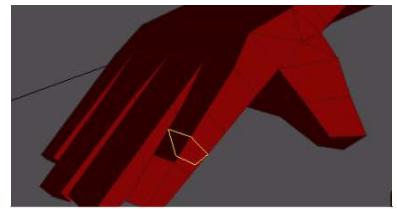


Abbildung 37: Loop-Cut (eigene Darstellung)

wurden konnten und die Hand ihre typische Form erhielt. Um den Hals zu formen, wurden die obersten Vertices im Nackenbereich zusammengerückt und extrudiert. Beim Kopf angelangt, wurde weiter extrudiert und skaliert, sodass eine sphärenartige Form entstanden ist.

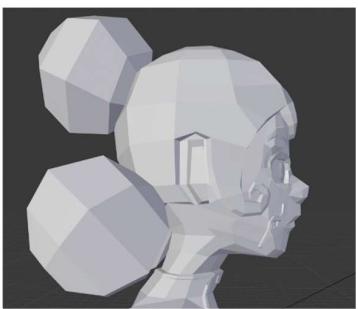


Abbildung 38: Details am Kopf des Modells (eigene Darstellung)

Aus dem Grundgerüst mussten die Details wie die Ohren, die Nase oder die Sommersprossen erstellt werden. Mit Hilfe des Knife-Tools konnten beispielsweise die Ohren oder die Haare aus dem Mesh herausgeschnitten werden. Dem Gesicht wurden ebenfalls durch Schnitte im Mesh mehr Vertices zur Modellierung zur Verfügung gestellt. Es wurde immer wieder zwischen den verschiedenen Details gewechselt und die Positionen der Vertices wurden angepasst, denn durch

Verschiebungen wurden andere Teile des Mesh verändert, welche wiederhergestellt werden mussten, indem an dieser Stelle eine weitere Änderung vorgenommen wurde.

Die wuscheligen Haare basieren ebenfalls auf Zylindern. Das Mesh des Kopfs erhielt an den Schnittstellen mit dem Knife-Tool Schnitte, die Vertices der beiden Elemente wurden einzeln verschmolzen. Die Pupillen der Augen wurden aus der Grundfläche eines separat hinzugefügten Zylinders geformt und wie die Arme mit dem bestehenden Körper zu einem Objekt gemacht, jedoch schweben sie auf dem restlichen Körper, da sie nicht mit diesem verschmolzen wurden. Die Details auf dem Anzug wurden ebenfalls wie die Elemente im Gesicht und am Kopf erstellt.

Mit dem erstellten Cel Shader, dessen Erstellungsprozess im Unterkapitel „Cel Shader in Blender“ erläutert wird, wurden Materials erstellt, die den einzelnen Flächen des Modells zugewiesen wurden. Das Modell erhielt so seine Farben, welche stets verstellbar sind. Gewisse Teile des Modells wurden zudem unter Verwendung eines Emission Node zum Leuchten gebracht.

Nachdem dem Mesh der letzte Feinschliff verpasst wurde, konnte der Mirror-Modifier aufs Modell appliziert werden. Durch diesen wurde der Mesh-Klon zu einem richtigen Mesh. Durch die Merge-Option wurden die Vertices auf der Spiegelachse automatisch miteinander verschmolzen, das Mesh besass somit keine Zwischenräume.

Der Charakter musste posiert werden und benötigte somit ein Rig. Es wurde ein Bone im unteren Bereich des Torsos eingefügt. Der X-Axis Mirror wurde angeschaltet, damit die Bones symmetrisch geklont wurden. Die Knochen wurden



Abbildung 39: Leuchtende Materials (eigene Darstellung)

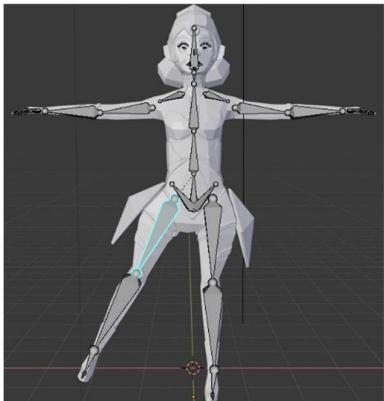


Abbildung 41: Mit Hilfe des Rig bewegtes Bein (eigene Darstellung)

muss. Die Bones erhielten das Mesh und somit ihren Einflussbereich durch automatische Weight-Paint-Zuweisung.

Allfällige Fehler wurden im Weight Paint Mode nachbearbeitet. Das Mesh des Modells wurde in einer T-förmigen Pose erstellt, sodass sämtliche Bereiche am Körper mit dem Weight-Paint-Pinsel leicht zu erreichen waren.

extrudiert und so platziert, dass alle Bereiche, die von einer Bewegung beeinflussbar sein sollten, mindestens einen Knochen erhielten, der von Gelenk zu Gelenk reicht. Das schmale Ende stellte dabei immer das Ende, das breite den Anfang, dar. Die Hände erhielten für jeden Fingerknochen jeweils einen Bone, damit sie wie echte Hände bewegbar wurden. Die Anzahl an Bones richtete sich stets danach, wie beweglich die entsprechende Stelle sein

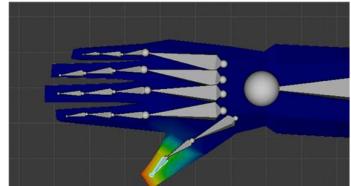


Abbildung 40: Weight Painting des Daumens (eigene Darstellung)

Nach der Fertigstellung des Modells wurde dieses mit Hilfe des Pose Mode posiert. Erst musste dazu die Kamera in Position gebracht werden, um anschliessend Bilder aus diversen Perspektiven zu erhalten, indem diese in Eevee gerendert wurden. Dazu wurde stetig zwischen der Kameraansicht und dem Ansichtsmodus des 3D-Viewports gewechselt.

3.3 Erstellung eines Cel Shader in Blender

3.3.1 Zusammenhänge der Nodes

Die Nodes arbeiten im in der Abbildung 42 und **Fehler! Verweisquelle konnte nicht gefunden werden.** abgebildeten Shader von links nach rechts und nehmen in jedem Node einen Wechsel vom Input zum Output vor. Die Normale ist ein Vektor, welcher senkrecht auf die Oberfläche gerichtet ist. Die Normalen können durch eine Normal Map manipuliert werden. Der Geometry Node liefert verschiedene Informationen zur Geometrie des Mesh, wie zum Beispiel die Normalen. Die Normale wird in den Shader Node eingegeben und dieser berechnet, wie sich das Licht verhält. Der Diffuse Node kann nur berechnen, wie das Licht auf der Oberfläche des Objekts aufprallen und streuen soll. Der Specular Node berechnet nur die Reflexionen und die Glanzpunkte. Die Diffuse und Specular Nodes liefern die Daten gebündelt im BSDF-Output. Der Shader To RGB Node extrahiert daraus die Farbwerte. Der ColorRamp Node nimmt die generierten Eingaben vom Specular Node und macht die Glanzpunkte weiss und den Rest der Oberfläche schwarz, er verwandelt also den glatten Verlauf in ein scharfes Bild. Für Diffuse-Node-Anwendungen wird der ColorRamp Node so eingestellt, dass die Schatten grau werden, die Mitteltöne hellgrau und die am stärksten beleuchteten Stellen weiss. Der Group Input wird verwendet, um die Farbe der Glanzpunkte vom Specular Node und die Farbe des Objekts zu erhalten, die mit dem ColorRamp-Output über den Multiply Node kombiniert werden. Und schliesslich werden beide Outputs, welche die Glanzpunkte und Schattierungen ausgeben, mit dem Add Node zu einer Einheit kombiniert.

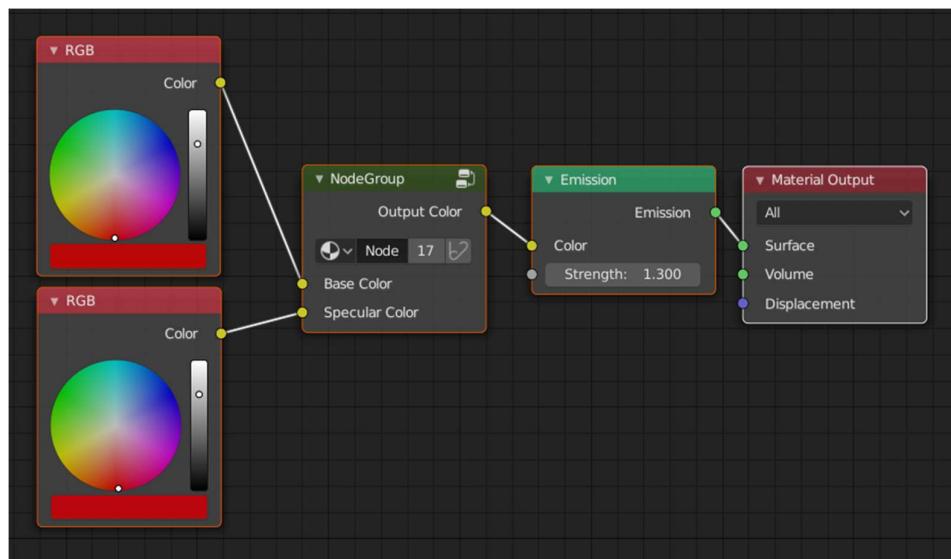


Abbildung 42: Cel Shader in Blender, Node Group (eigene Darstellung)

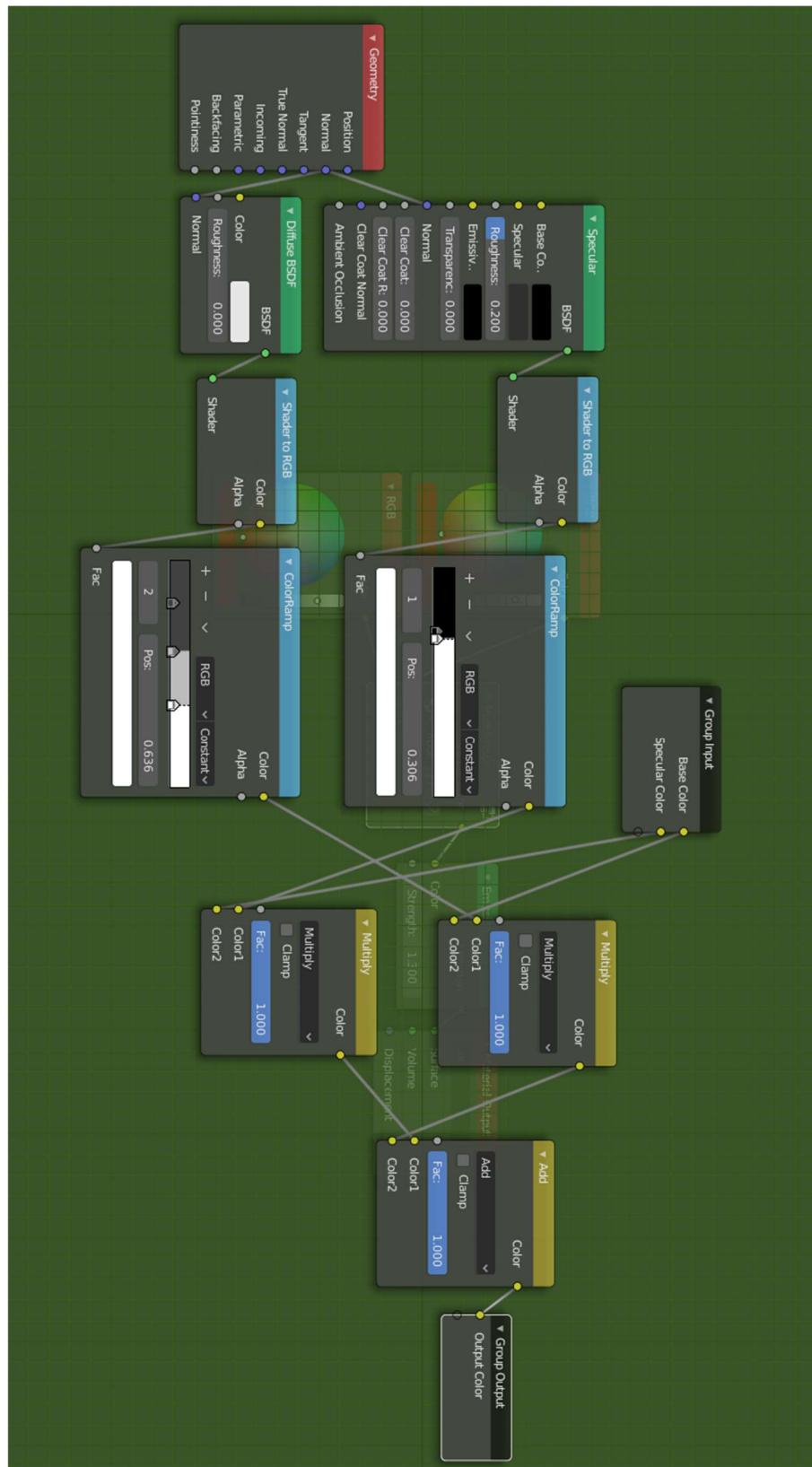


Abbildung 43: Inhalt der Node Group, Cel Shader in Blender (eigene Darstellung)

3.3.2 Pseudocode des Cel Shader

Zur Veranschaulichung der Funktionsweise der Nodes im erstellten Cel Shader wurde ein Pseudocode geschrieben.

Definitionen Vektoroperationen

dot(a, b): Das Skalarprodukt von Vektor a und b

a + b: Komponentenweise Addition $(x_1, y_1) + (x_2, y_2) = (x_1 + x_2, y_1 + y_2)$

a - b: Komponentenweise Subtraktion $(x_1, y_1) - (x_2, y_2) = (x_1 - x_2, y_1 - y_2)$

a * b: Komponentenweise Multiplikation $(x_1, y_1) * (x_2, y_2) = (x_1 * x_2, y_1 * y_2)$

Definitionen Operationen

a^b: a hoch b

:= : Zuweisung einer Variabel

Diffuse

input:

Normal: (x,y,z) // kommt von der Geometry-Node

inputBlender:

L: (x,y,z) // kennt der Diffuse BSDF Shader

DiffuseColor: (r,g,b) // ist einstellbar im Node, Farbkonstante

LightColor: (r,g,b) // kennt der Specular BSDF Shader

IL := LightColor

dotNL := dot(Normal, L)

ID := IL * DiffuseColor * dotNL

output:

ID: (r,g,b)

Specular

input:

Normal: (x,y,z) // kommt vom Geometry Node

inputBlender:

L: (x,y,z) // kennt der Specular BSDF Shader

EP: (x,y,z) // kennt der Specular BSDF Shader

SpecularColor: (r,g,b) // ist einstellbar im Node, Farbkonstante

Roughness // ist einstellbar im Node, Zahl zwischen 0 und 1

LightColor: (r,g,b) // kennt der Specular BSDF Shader

IL := LightColor

k := Roughness

r := 2 * (1 * Normal) * Normal - 1 // mit der Reflektionsformel berechnet

IS := IL * SpecularColor * dot(r, EP)^k

output:

IS: (r,g,b)

ColorRamp

```
input:  
    fac // Zahl zwischen 0 und 1  
  
inputBlender:  
    color1: (r,g,b)  
    color2: (r,g,b)  
    edge // Zahl zwischen 0 und 1  
  
if (fac > edge)  
    outcol := color2  
else  
    outcol := color1  
  
output:  
    outcol: (r,g,b)
```

Alles

```
input:  
    Normal: (x,y,z)  
    BaseColor: (r,g,b) // es werden Grautöne verwendet  
    SpecularColor: (r,g,b) // es werden Grautöne verwendet  
  
    spec := Specular(Normal)  
    diff := Diffuse(Normal)  
    diffRamp := ColorRamp(diff.r) // weil bei Grautönen nur ein Kanal benötigt  
    wird  
    specRamp := ColorRamp(spec.r) // weil bei Grautönen nur ein Kanal benötigt  
    wird  
  
    diffColor := diffRamp * BaseColor  
    specColor := specRamp * SpecularColor  
  
    OutputColor := diffColor + specColor  
  
output:  
    OutputColor: (r,g,b)
```

4 Resultate des Erstellungsprozesses

Das Produkt in Form eines 3D-Modells mit Cel Shaders, welches von Grund auf selbst entwickelt wurde, wird im Folgenden in diversen Posen abgebildet. Da es sich um einen Low-Poly-Charakter handelt, wurden so wenige Polygonnetze wie möglich bei seiner Erstellung verwendet, was ihm seinen eckigen Stil verleiht. Das Modell enthält ein Rig und ist somit beweglich. Der Cel Shader sorgt für klare, comichafte Farbgrenzen. Das erstellte Hotkey-Sheet resultierte aus den Erfahrungen mit Blender und fasst die wichtigsten Tastenkombinationen zusammen.



Abbildung 45: Render des Modells in T-Pose
(eigene Darstellung)

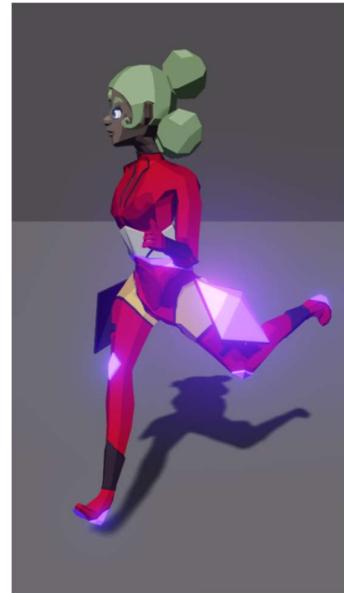


Abbildung 44: Render des Modells
in Renn-Pose (eigene Darstellung)



Abbildung 46: Zweiter Render des Modells in rennender Pose (eigene
Darstellung)



Abbildung 47: Render des Modells in ängstlicher Pose (eigene Darstellung)



Abbildung 48: Render des Modells in verlegener Pose (eigene Darstellung)

5 Diskussion

Eines der Ziele dieser Maturaarbeit war es, innerhalb eines eingeschränkten Zeitraums den Umgang mit Blender 2.8 und dessen UI zu erlernen. Dieses Ziel wurde vollständig erreicht, es konnte im betrachteten Bereich des 3D-Modeling ein Entwicklungsprozess von Anfängern zu Experten durchlaufen werden. Zu Beginn der Maturaarbeit war der Umgang mit den Funktionen des Programms Blender recht fremd, jedoch konnte dies mit Hilfe von Literatur und online verfügbaren Tutorials geändert werden. Allerdings bezogen sich nicht alle Tutorials auf die Version 2.8, was besonders zu Beginn des Prozesses zu Schwierigkeiten führte, denn verglichen mit älteren Versionen weist die verwendete Version andere Shortcuts und Bezeichnungen auf. Schlüsselbegriffe via Google und im offiziellen Blender-Manual zu suchen, erwies sich hierbei als sehr hilfreich, denn besonders im Manual werden die Begriffe ausführlich erklärt. So wurden die wichtigsten Funktionen und Werkzeuge, um ein 3D-Modell erstellen zu können, erlernt und sind in der Arbeit sachgemäß wiedergegeben.

In dieser eingeschränkten Zeit konnte natürlicherweise nicht das ganze User Interface mit sämtlichen Tools, die Blender zu bieten hat, erlernt werden. Das Endprodukt in Form eines 3D-Charakters im Low-Poly-Stil wurde erfolgreich erstellt, denn es weist sämtliche Merkmale dieses Stils auf. Auch wenn der Begriff Low Poly ein relativ ist, kann behauptet werden, dass das Ziel klar erreicht wurde, denn das Mesh des Modells besitzt in der Gesamtanzahl 4'895 Tris, also Dreiecksflächen. Im Vergleich mit dem 3D-Modell des Helden Link aus „The Legend of Zelda: Breath of the Wild“, ein Videospiel welches 2017 erschien, sieht die Situation anders aus. Mit 12'541 Tris verglichen ist das Endprodukt dieser Maturaarbeit ein Low Poly Modell [59]. Umso mehr wird diese Behauptung unterstützt, wenn das Endprodukt aus einem künstlerischen Blickwinkel betrachtet wird, denn die Kanten und Ecken sind im 3D-Modell erkennbar. Die theoretischen Grundlagen, wie das Erstellen des Cel-Shaders, wurden einerseits mit Hilfe von Tutorials, andererseits auch durch zahlreiche Eigenversuche erforscht, was in einem funktionsfähigen Shader resultierte, denn das Modell besitzt klare Grenzen zwischen den hellen und dunklen Stellen. Durch die Erstellung des 3D-Modells wurden viele Tastenkombinationen erlernt, welche nun reflexartig getätigten werden, wodurch sich der Erstellungsprozess beschleunigt. Zu den kontinuierlich verwendeten Tastenkombinationen wurde ein Hotkey-Sheet erstellt, welches als Spick dienen kann.

Die Erkenntnisse und mathematischen Formeln wurden in eigenen Skizzen und mit eigens erstellten Beispielen dargelegt, wobei diese nur mit Hilfe des erlangten Wissens und Know-hows erstellt werden konnten, zum Beispiel, wie Glanzpunkte und Schatten auf einem 3D-

Modell erzeugt werden können. Unter Verwendung eines Rig konnte das erstellte Mesh zum Leben erweckt und in Szene gesetzt werden. Durch die Bewegungen der Bones konnte sichergestellt werden, dass sich keine Lücken im Mesh befinden, denn diese wären durch ein Loch, in welches man hineinblicken hätte können, sichtbar geworden. Das Rig diente somit nicht nur zur Erstellung der Bilder im Kapitel „Resultate des Erstellungsprozesses“, sondern auch der Kontrolle.

Durch die Wahl der Render-Engine Eevee konnte das Modell in Echtzeit betrachtet werden, wodurch es in einem Spiel implementierbar ist. Eevee hat allerdings einige Grenzen, denn aufgrund der Annäherung an die physikalisch korrekte Lichtberechnung bei der Schattendarstellung müssen situationsbedingt weitere Lichtquellen in der Szene platziert werden. Ohne diese weiteren Lampen wäre der Charakter hinter einem transparenten Objekt unbeleuchtet geblieben.

Wäre hingegen Cycles verwendet worden, so könnten lediglich Bilder des Modells gerendert werden, aber keine Bewegungen in Echtzeit erzeugt und betrachtet werden. Die Schatten wären jedoch realistischer dargestellt und es wären keine Umstände nötig, um ein Objekt hinter einer transparenten Ebene zusätzlich zu beleuchten. Es ist nicht möglich, zu behaupten, dass die neuere Render-Engine Eevee Cycles ersetzt, denn beide haben ihre Stärken und Schwächen und dienen entsprechend unterschiedlichen Zwecken. Sind realistische Schatten nicht relevant, so wird Eevee aufgrund seiner schnellen Rendergeschwindigkeit bevorzugt. Sollte das Bild so realitätsnah wie möglich erscheinen, wird zu Cycles geraten.

6 Schluss

Zusammenfassung der Arbeit

In dieser Maturaarbeit wurde der Themenbereich der 3D-Modellierung ausführlich betrachtet, um ein eigenes 3D-Modell mit einem eigenen Shader zu entwerfen und zu erstellen. In der Theorie wurden das Programm Blender, die Funktionsweise von Shaders und Low Poly näher betrachtet. Es wurden die für die Erstellung eines 3D-Modells in Blender relevanten Funktionen und Tools notiert und herausgearbeitet. Der Erstellungsprozess wurde dokumentiert und zur Präsentation des Resultats wurden Bilder des erstellten Charakters gerendert.

Sachliches Fazit

Die Erstellung eines beweglichen 3D-Modells und des Shaders waren erfolgreich, es konnte das gewünschte visuelle Resultat erzielt werden. Das Hotkey-Sheet fasst die wichtigsten verwendeten Tastenkombinationen zusammen und kann neuen Benutzern den Einstieg in Blender erleichtern.

Erfahrungen und Ausblicke

Bei der Themenwahl hatten wir keine grundlegenden Probleme, da wir uns beide für ähnliche Themengebiete interessieren und uns deshalb auch schnell für ein für uns passendes und interessantes Thema entscheiden konnten. Als wir zu Beginn der Maturaarbeit intensiv recherchierten, stiessen wir auf das Programm Blender, mit dem atemberaubende 3D-Modelle erstellt werden können. Diese Faszination führte schliesslich zur Themenwahl: Wir wollten unsere eigenen Kunstwerke schaffen können, waren aber auch neugierig darauf, wie der Computer diese dreidimensionalen Kreationen eigentlich darstellt.

Durch Voice Chats und Screen Sharing via Discord und das Hochladen der Dateien über GitHub und Discord funktionierte unsere Zusammenarbeit ohne Probleme. Da wir in der Vergangenheit bereits gemeinsam an diversen Projekten gearbeitet hatten, waren wir ein eingespieltes Team. So konnten die Aufgaben so verteilt werden, dass jede von uns ihre Stärken nutzen konnte.

Blender in einer begrenzten Zeit zu erlernen, war für uns eine neue Herausforderung, die wir allerdings gemeistert haben, denn wir haben es geschafft, ein zweidimensionales Konzept in den dreidimensionalen Raum zu bringen. Allerdings dauerte das Erstellen eines Objekts zu Beginn der Übungs- und Einarbeitungsphase lange, daher würden wir in einer

weiterführenden Arbeit mit Blender die Hotkeys frühzeitig auswendig lernen, um den Workflow zu beschleunigen, denn das Anklicken der Icons in Blender kostet immens Zeit.

Das erstellte 3D-Modell könnte weiterführend verwendet werden, indem es in Blender animiert wird, in Unity verwendet und so beispielsweise in einem Videospiel implementiert wird. Es gibt aber auch Webseiten, wie <https://sketchfab.com/>, auf welche man das kreierte Modell hochladen kann. So kann dieses von anderen Nutzern betrachtet werden. Das Werk könnte so auch verkauft werden. Das Mesh könnte aber auch weiterbearbeitet werden und so mit Hilfe eines Modifiers in ein High-Poly-Modell verwandelt werden.

7 Glossar

A

A in RGB A: Der Alphakanal oder α -Kanal ist ein zusätzlicher Kanal, der in Rastergrafiken zusätzlich zu den Farbinformationen die Transparenz (Durchsichtigkeit) der einzelnen Pixel (Bildpunkte) speichert. Die Darstellung eines Bildes mit Alphakanal auf einem Hintergrund wird als Alpha Blending bezeichnet. 19

Ambient: Eine Umgebungslichtquelle stellt eine omnidirektionale, lichtstarke und farbfeste Lichtquelle dar, die alle Objekte in der Szene gleichermassen betrifft. 19

Animation: Animation ist im engeren Sinne jede Technik, bei der durch das Erstellen und Anzeigen von Einzelbildern für den Betrachter ein bewegtes Bild geschaffen wird. 1, 3

API: Eine Programmierschnittstelle (auch Anwendungsschnittstelle, genauer Schnittstelle zur Programmierung von Anwendungen), häufig nur kurz API genannt (von englisch application programming interface, wörtlich „Anwendungs-programmier-schnittstelle“), ist ein Programmteil, der von einem Softwaresystem anderen Programmen zur Anbindung an das System zur Verfügung gestellt wird. 5

B

Betriebssystem: Ein Betriebssystem, auch OS (von englisch operating system) genannt, ist eine Zusammenstellung von Computerprogrammen, die die Systemressourcen eines Computers wie Arbeitsspeicher, Festplatten, Ein- und Ausgabegeräte verwaltet und diese Anwendungsprogrammen zur Verfügung stellt. 4

Bit: Der Begriff Bit wird als Masseinheit für den Informationsgehalt verwendet, wobei 1 Bit der Informationsgehalt, der in einer Auswahl aus zwei

gleich wahrscheinlichen Möglichkeiten enthalten ist. 33

BSDF: Diese Abkürzung steht für Bidirectional scattering distribution function und beschreibt die allgemeine mathematische Funktion, die die Art und Weise beschreibt, wie das Licht von einer Oberfläche gestreut wird. 27, 28, 30, 41

C

C: C ist eine imperative und prozedurale Programmiersprache, die der Informatiker Dennis Ritchie in den frühen 1970er Jahren an den Bell Laboratories entwickelte. 5

C++: C++ ist eine von der ISO genormte Programmiersprache. C++ ermöglicht sowohl die effiziente und maschinennahe Programmierung als auch eine Programmierung auf hohem Abstraktionsniveau. 5

Compiler: Ein Compiler ist ein Computerprogramm, das Quellcodes einer bestimmten Programmiersprache in eine Form übersetzt, die von einem Computer (direkt) ausgeführt werden kann. 5

Computergrafik: Die Computergrafik ist ein Teilgebiet der Informatik, das sich mit der computergestützten Bilderzeugung, im weiten Sinne auch mit der Bildbearbeitung befasst. Mit den Mitteln der Computergrafik entstandene Bilder werden Computergrafiken genannt. 19, 32

Cosinus: Sinus- und Kosinusfunktion (auch Cosinusfunktion) sind elementare mathematische Funktionen. 20, 21, 22, 23, 25

CUDA: CUDA (früher auch Compute Unified Device Architecture genannt) ist eine von Nvidia entwickelte Programmier-Technik, mit der Programmteile durch den Grafikprozessor (GPU) abgearbeitet werden können. 16

Cyberpunk: Im Unterschied zu den klassischen Utopien vieler anderer Science-Fiction-Genres ist

die Welt des Cyberpunk nicht glänzend und steril-sauber, sondern düster, und von Gewalt und Pessimismus geprägt. 35

F

Fliesskommazahl: Eine Gleitkommazahl – häufig auch Fliesskommazahl genannt (englisch floating point number oder kurz float, wörtlich Zahl mit flottierendem Punkt oder auch [wohl weiter lehnübersetzt] Gleitpunktzahl) – ist eine angenäherte Darstellung einer reellen Zahl. 33

Fragment: In der Computergrafik ist ein Fragment die Daten, die notwendig sind, um den Wert eines einzelnen Pixels im Frame-Puffer zu erzeugen. 16, 18, 19, 20

G

Gizmo: Gizmo (engl. Platzhalterwort, etwa „Ding“, „Dingens“, „Dingsbums“) steht im englischen Sprachgebrauch auch für ein Gadget. 8, 13

H

Hotkeys: Als Tastenkombination (auch Tastaturkombination, Tasturbefehl, Tastatkürzel, Tastenkürzel, Tastensequenz, Hotkey, Shortcut) wird das gleichzeitige oder aufeinanderfolgende Drücken mehrerer Tasten auf Computertastaturen in einer bestimmten Reihenfolge bezeichnet. 10

I

interpoliert: In der numerischen Mathematik bezeichnet der Begriff Interpolation (aus lateinisch inter = dazwischen und polire = glätten, schleifen) eine Klasse von Problemen und Verfahren. Zu gegebenen diskreten Daten (z. B. Messwerten) soll eine stetige Funktion (die sogenannte Interpolante oder Interpolierende) gefunden werden, die diese Daten abbildet. Man sagt dann, die Funktion interpoliert die Daten. 18

Interpreter: Als Interpreter wird ein Computerprogramm bezeichnet, das eine Abfolge von Anweisungen scheinbar direkt ausführt, wobei das Format der Anweisungen vorgegeben ist. Der Interpreter liest dazu eine oder mehrere Quelldateien ein, analysiert diese und führt sie anschliessend Anweisung für Anweisung aus, indem er sie in Maschinencode übersetzt, die ein Computersystem direkt ausführen kann. Interprete sind deutlich langsamer als Compiler, bieten im Allgemeinen jedoch eine bessere Fehleranalyse. 6

L

Lambertian: Das Lambertsche Gesetz (auch Lambertsches Kosinusgesetz) beschreibt, formuliert von Johann Heinrich Lambert, wie durch den perspektivischen Effekt die Strahlungsstärke mit flacher werdendem Abstrahlwinkel abnimmt. 30

M

Mesh: Ein Mesh in der Geometrie ist eine lückenlose und überlappungsfreie Partition eines Bereichs des Raumes durch eine Menge von Gitterzellen. 12

Motion Capture: Unter Motion Capture, wörtlich Bewegungs-Erfassung, versteht man ein Tracking-Verfahren zur Erfassung und Aufzeichnung von Bewegungen, so dass Computer diese wiedergeben, analysieren, weiterverarbeiten und zur Steuerung von Anwendungen nutzen können. 3

N

Normalenvektor: In der Geometrie ist ein Normalenvektor, auch Normalvektor, ein Vektor, der orthogonal (d. h. rechtwinklig, senkrecht) auf einer Geraden, Kurve, Ebene, (gekrümmten) Fläche oder einer höherdimensionalen Verallgemeinerung eines solchen Objekts steht. In der Geometrie ist ein Normalenvektor, auch Normalvektor, ein Vektor, der orthogonal (das heisst rechtwinklig,

senkrecht) auf einer Geraden, Kurve, Ebene, (gekrümmten) Fläche oder einer höherdimensionalen Verallgemeinerung eines solchen Objekts steht. 20

O

OpenCL: OpenCL (englisch Open Computing Language) ist eine Schnittstelle für uneinheitliche Parallelrechner, die z. B. mit Haupt-, Grafik- oder digitalen Signalprozessoren ausgestattet sind. 16

OpenGL: OpenGL (Open Graphics Library; deutsch Offene Grafikbibliothek) ist eine Spezifikation einer plattform- und programmiersprachen-übergreifenden Programmierschnittstelle (API) zur Entwicklung von 2D- und 3D-Computergrafikanwendungen. Der OpenGL-Standard beschreibt etwa 250 Befehle, die die Darstellung komplexer 3D-Szenen in Echtzeit erlauben. 8

Open-Source: Als Open Source (aus englisch open source, wörtlich offene Quelle) wird Software bezeichnet, deren Quelltext öffentlich und von Dritten eingesehen, geändert und genutzt werden kann. Open-Source-Software kann meistens kostenlos genutzt werden. 3

Oren-Nayar: Das von Michael Oren und Shree K. Nayar entwickelte Reflexionsmodell Oren-Nayar ist ein Reflexionsmodell für die diffuse Reflexion von rauen Oberflächen. Es hat sich gezeigt, dass es das Aussehen einer Vielzahl von natürlichen Oberflächen wie Beton, Putz, Sand usw. genau vorhersagt. 30

P

Polygon: Ein Polygon erhält man, indem in einer Zeichenebene mindestens drei verschiedene (nicht kollineare) Punkte miteinander verbunden werden durch Strecken, Kanten oder Seiten genannt, sodass ein geschlossener Polygonzug mit ebenso vielen Ecken entsteht, beispielsweise ein Dreieck oder ein Viereck. 32

Prozessor: Ein Prozessor ist ein (meist sehr stark verkleinertes) (meist frei) programmierbares Rechenwerk, also eine Maschine oder eine elektronische Schaltung, die gemäß übergebenen Befehlen andere Maschinen oder elektrische Schaltungen steuert und dabei einen Algorithmus (Prozess) vorantreibt, was meist Datenverarbeitung beinhaltet. 5

Pseudocode: Pseudocode ist ein Programmcode, der nicht zur maschinellen Interpretation, sondern lediglich zur Veranschaulichung eines Paradigmas oder Algorithmus dient. Meistens ähnelt er höheren Programmiersprachen, gemischt mit natürlicher Sprache und mathematischer Notation. 44

Python: Python ist eine universelle, üblicherweise interpretierte höhere Programmiersprache. 5, 6

Q

Quellcode: Quelltext, auch Quellcode (englisch source code) oder unscharf Programmcode genannt, ist in der Informatik der für Menschen lesbare, in einer Programmiersprache geschriebene Text eines Computerprogrammes. 3

R

Rasterization: In der 2D-Computergrafik bezeichnet Rasterization, auch Rendern oder Scankonvertierung genannt, die Umwandlung einer Vektor- in eine Rastergrafik. 9

Raytracer: Raytracing (dt. Strahlverfolgung oder Strahlenverfolgung) ist ein auf der Aussendung von Strahlen basierender Algorithmus zur Verdeckungsberechnung, also zur Ermittlung der Sichtbarkeit von dreidimensionalen Objekten von einem bestimmten Punkt im Raum aus. 16

Rechenleistung: Die Rechenleistung (auch Datenverarbeitungsleistung oder Performanz, englisch computing power oder performance genannt) ist ein Mass für Rechenmaschinen und Datenverarbeitungs- (kurz DV-Systeme) oder

informationstechnische Systeme (kurz IT-Systeme). 33

rechthändiges Koordinatensystem: Als Rechtssystem bzw. rechtshändiges Koordinatensystem werden in der Mathematik und Physik gewisse Systeme (mit einer festgelegten Reihenfolge) von zwei Vektoren in der Ebene bzw. drei Vektoren im Raum bezeichnet. Ein Rechtssystem im dreidimensionalen Raum sind drei Vektoren x, y und z, wenn vom Endpunkt des Vektors z aus gesehen die Vektoren x, y ein Rechtssystem in der Ebene bilden. 17

Reflektionsformel: Angenommen sei ein Strahl, der aus der Richtung v (normiert) auf eine Oberfläche mit Normale r (auch normiert) trifft. Dann bestimmt sich die normierte Reflexions-Richtung durch die Formel: $\vec{r} = 2(\vec{n} \cdot \vec{l})\vec{n} - \vec{l}$ 22

Render: Rendern (von englisch (to) render, deutsch etwas machen oder auch etwas wiedergeben) bezeichnet in der Computergrafik die Erzeugung eines Bildes aus Rohdaten. 8, 9, 11, 13, 16

Render-Engine: Eine Render-Engine gibt die in der 3D-Welt vorhandenen Daten auf der Anzeige aus. 8

RGB: Ein RGB-Farbraum ist ein additiver Farbraum, der Farbwahrnehmungen durch das additive Mischen dreier Grundfarben (Rot, Grün und Blau) nachbildet. Das Farbsehen des Menschen ist von drei Zapfentypen geprägt. Dieser Farbraum basiert im Prinzip auf der Dreifarbentheorie. 28

S

Simulation: Die Simulation oder Simulierung ist eine Vorgehensweise zur Analyse von Systemen, die für die theoretische oder formelmäßige Behandlung zu komplex sind. 3

Skalarprodukt: Das Skalarprodukt (auch inneres Produkt, selten Punktprodukt) ist eine mathematische Verknüpfung, die zwei Vektoren eine Zahl (Skalar) zuordnet. 20, 21, 23, 25

Steampunk: Häufige Elemente des Steampunks sind dampf- und zahnradgetriebene Mechanik,

viktorianischer Kleidungsstil und ein viktorianisches Werte-Modell, eine gewisse Do-it-yourself-Mentalität und Abenteuerromantik. 35

T

Textur: In der Computergrafik verwendet man Texturen als „Überzug“ für 3D-Modelle, um deren Detailgrad zu erhöhen, ohne dabei jedoch den Detailgrad der Geometrie zu erhöhen. 25

U

Unity: Unity ist eine Laufzeit- und Entwicklungsumgebung für Spiele (Spiel-Engine) des Unternehmens Unity Technologies. 52

User Interface: User Interface; Die Benutzerschnittstelle oder auch Nutzerschnittstelle ist die Stelle oder Handlung, mit der ein Mensch mit einer Maschine in Kontakt tritt. 8, 10

V

Vertex: In der 3D-Computergrafik ist ein Vertex ein Eck- bzw. Scheitelpunkt eines Primitivs und enthält neben einer Positionsangabe in Form eines 3D-Vektors meistens noch andere Angaben wie zum Beispiel eine Farbe, Transparenz oder eine zweite Positionsangabe, die für andere Zwecke verwendet werden kann (z. B. Texturkoordinaten). Die meisten Daten der Vertices werden durch das Shading über die zugehörigen Primitive interpoliert.

In der polygonalen Modellierung ist der Vertex ein Grundbaustein: Zwei Vertices ergeben zusammen die Endpunkte einer Linie, drei Vertices definieren ein Dreieck usw. 16, 17, 18

Viewport: Als Viewport (übersetzt etwa Sichtöffnung) wird je nach Anwendungsgebiet ein Ausschnitt eines Bildes, eines Videos, einer virtuellen oder realen Welt bezeichnet oder der für die Darstellung zur Verfügung stehende Bereich. 8, 12, 13

8 Literaturverzeichnis

- [1] HEISE MEDIEN GMBH & Co. KG, Blender 2.80,
<https://www.heise.de/download/product/blender-13137>, besucht: 12.07.2019.
- [2] BLENDER FOUNDATION, About, <https://www.blender.org/about/>, besucht: 17.08.2019.
- [3] WIKIPEDIA, Blender (Software), [https://de.wikipedia.org/wiki/Blender_\(Software\)](https://de.wikipedia.org/wiki/Blender_(Software)),
besucht: 12.07.2019.
- [4] VAN GUMSTER, Jason, 2015, Blender for Dummies, 3. Auflage, USA, John Wiley & Sons, Inc..
- [5] FELINTO, Dalai, Pan, Mike, 2014, Game Development with Blender, Canada, Cengage Learning PTR.
- [6] BLENDER, Download Blender 2.80, <https://www.blender.org/download/>, besucht:
30.07.2019.
- [7] WIKIPEDIA, Blender Foundation, https://de.wikipedia.org/wiki/Blender_Foundation,
besucht: 13.07.2019.
- [8] BLENDER ANIMATION STUDIO, 2019, Spring - Blender Open Movie,
<https://www.youtube.com/watch?v=WhWc3b3KhnY>, besucht am 5.10.2019.
- [9] SCHERCH, Christiane, Was ist CPU? Einfach erklärt, 2019,
https://praxistipps.chip.de/was-ist-cpu-einfach-erklaert_41813, besucht: 17.07.2019.
- [10] SAWAKINOME, Was ist der Unterschied zwischen Skriptsprache und
Programmiersprache?, <https://www.sawakinome.com/articles/technology/what-is-the-difference-between-scripting-language-and-programming-language.html>, besucht:
05.07.2019.
- [11] ITPEDIA, 2017, Skriptsprache ist etwas anderes als Programmiersprache,
<https://de.itpedia.nl/2018/07/24/scripttaal-is-iets-anders-dan-programmeertaal/>,

- besucht 09.07.2019.
- [12] BLENDER FOUNDATION, 2019, Python API Overview,
https://docs.blender.org/api/current/info_overview.html, besucht: 08.08.2019.
- [13] GRAPHIC PIZIADAS, 2014, Programmieren mit Python in Blender,
<http://piziadas.com/de/2015/10/programando-con-python-en-blender.html>, besucht: 08.08.2019.
- [14] BLENDER FOUNDATION, 2019, Blender, made by you,
<https://www.blender.org/download/releases/2-80/>, besucht: 12.08.2019.
- [15] RGB-LABS.COM, 2018, Blender 127 – Blender 2.8 kommt. Bald! [Update],
<https://www.rgb-labs.com/blender-127-blender-2-8-kommt-bald/>, besucht: 12.08.2019.
- [16] LAMPEL, Johnathan, 2019, Top 5 Changes in Blender 2.8,
<https://cgcookie.com/articles/top-5-changes-coming-with-blender-2-8>, besucht: 12.08.2019.
- [17] VISCIRCLE, Was Sie alles über Eevee wissen sollten., <https://viscircle.de/was-sie-alles ueber-eevee-wissen-sollten/>, besucht: 12.08.2019.
- [18] LAMPEL, Jonathan, 2019, Cycles vs. Eevee - 15 Limitations of Real Time Rendering in Blender 2.8, <https://cgcookie.com/articles/blender-cycles-vs-eevee-15-limitations-of-real-time-rendering>, besucht: 08.08.2019.
- [19] HOFMANN, Gottfried, 2019, 3D-Renderer Blender 2.8: Schnelle Render-Engine, renovierte Oberfläche, <https://www.heise.de/newsticker/meldung/3D-Renderer-Blender-2-8-Schnelle-Render-Engine-renovierte-Oberflaeche-4479103.html>, besucht: 14.08.2019.
- [20] WILK, Grant, 2018, Get Ready For Eevee, Blender's New Real-Time Rendering Engine, <https://cgcookie.com/articles/get-ready-for-eevee-blender-s-new-real-time-rendering-engine>, besucht: 16.08.2019.
- [21] WIKIPEDIA, OpenGL, <https://en.wikipedia.org/wiki/OpenGL>, besucht: 12.09.2019.

- [22] TU-ILMENAU.DE, Computergrafik II, https://www.tu-ilmenau.de/fileadmin/media/gdv/Lehre/Computergrafik_II/Uebung/Computergrafik_II_UEbung_1_Folien.pdf, besucht: 10.09.2019.
- [23] ANDREUSSI, Francesco, 2018, Shading with GLSL, https://www.uni-weimar.de/fileadmin/user/fak/medien/professuren/Computer_Graphics/CG_WS_18_19/CG/03_Shaders.pdf, besucht: 09.09.2019.
- [24] OPENGL WIKI, Vertex Shader, https://www.khronos.org/opengl/wiki/Vertex_Shader, besucht: 17.08.2019.
- [25] NVIDIA, Vertex Shaders, https://www.nvidia.com/object/feature_vertexshader.html, besucht: 13.09.2019.
- [26] WIKIPEDIA, Vertex-Shader, <https://de.wikipedia.org/wiki/Vertex-Shader>, besucht: 10.09.2019.
- [27] SCIAENGINEER, Globales Koordinatensystem, https://help.scia.net/15.0/de/rb/basics/global_co_ordinate_system.htm, besucht: 10.09.2019.
- [28] LIGHTHOUSE.3D.COM, GLSL Tutorial – Fragment Shader, <https://www.lighthouse3d.com/tutorials/glsl-tutorial/fragment-shader/>, besucht: 06.09.2019.
- [29] WIKIPEDIA, Pixel-Shader, <https://de.wikipedia.org/wiki/Pixel-Shader>, besucht: 07.09.2019.
- [30] INF.FU-BERLIN.DE, Phong Shading, http://www.inf.fu-berlin.de/lehre/WS06/19605_Computergrafik/doku/rossbach_siewert/phong.html, besucht: 16.09.2019.
- [31] PROF. SCHLECHTWEG, Stefan, Computergraphik Grundlagen, <http://www.mttcs.org/Skripte/Pra/Material/vorlesung7.pdf>, besucht: 15.09.2019.
- [32] LEARNOPENGL, Basic Lighting, <https://learnopengl.com/Lighting/Basic-Lighting>,

- besucht: 15.09.2019.
- [33] GEBHARDT, Nikolaus, Einige BRDF Modelle,
<http://www.irrlicht3d.org/papers/BrdfModelle.pdf>, besucht: 17.09.2019.
- [34] ROYSTAN, Toon Shader using Unity engine 2018.3, <https://roystan.net/articles/toon-shader.html>, besucht: 15.09.2019.
- [35] LEARN WEBGL, 2016, Specular Lighting,
http://learnwebgl.brown37.net/09_lights/lights_specular.html#glossary, besucht: 17.08.2019.
- [36] THE VIDEO GAME ALMANAC, 2019, The Legend Of Zelda: The Wind Waker | Retro Rated, <https://vgalmanac.com/reviews/the-legend-of-zelda-the-wind-waker-retro-rated/>, besucht: 30.09.2019.
- [37] WIKIPEDIA, Cel Shading, https://de.wikipedia.org/wiki/Cel_Shading, besucht: 03.09.2019.
- [38] GAMESTAR REDAKTION, 2009, Cel-Shading-Spiele - Die Technik erklärt, die Spiele vorgestellt, <https://www.gamestar.de/artikel/cel-shading-spiele-die-technik-erklaert-die-spiele-vorgestellt,2310172,seite2.html>, besucht: 05.09.2019.
- [39] SAVE EVERY UNIVERSE, 2019, Art Styles In Games, Borderlands, and Cel Shading, <https://anchor.fm/save-every-universe/episodes/Art-Styles-In-Games--Borderlands--and-Cel-Shading-e3kept>, besucht: 07.08.2019.
- [40] OESER, Florian, Simple Lighting, <http://www.florian-oeser.de/rtr/ue3/3.1.html>, besucht: 05.09.2019.
- [41] VIZUA, Cel-Shading/ Toon-Shading, <https://vizua.co/3d-glossar/cel-shading-toon-shading/>, besucht: 15.09.2019.
- [42] WIKIBOOKS, Blender Dokumentation: Nodes,
https://de.wikibooks.org/wiki/Blender_Dokumentation:_Nodes, besucht: 14.09.2019.
- [43] BLENDER 2.80 MANUAL, Node Groups,

https://docs.blender.org/manual/en/latest/editors/texture_node/types/groups.html?highlight=node%20groups, besucht: 13.09.2019.

- [44] BLENDER 2.80 MANUAL, Geometry Node,
https://docs.blender.org/manual/en/latest/render/shader_nodes/input/geometry.html, besucht 15.09.2019.
- [45] BLENDER 2.80 MANUAL, Specular BSDF,
https://docs.blender.org/manual/en/latest/render/shader_nodes/shader/specular_bsdf.html?highlight=specular%20bsdf, besucht: 15.09.2019.
- [46] BLENDER 2.80 MANUAL, Shader To RGB,
https://docs.blender.org/manual/en/latest/render/shader_nodes/converter/shader_to_rgb.html?highlight=shader%20rgb, besucht: 15.09.2019.
- [47] BLENDER 2.80 MANUAL, Color Ramp Node,
https://docs.blender.org/manual/en/latest/render/shader_nodes/converter/color_ramp.html?highlight=color%20ramp, besucht: 14.09.2019.
- [48] BLENDER 2.80 MANUAL, Multiply,
https://docs.blender.org/manual/en/latest/video_editing/sequencer/strips/effects/multiply.html?highlight=multiply, besucht: 16.09.2019.
- [49] BLENDER 2.80 MANUAL, Mix Node,
https://docs.blender.org/manual/en/latest/render/shader_nodes/color/mix.html?highlight=mix%20node, besucht: 15.09.2019.
- [50] BLENDER 2.80 MANUAL, Diffuse BSDF,
https://docs.blender.org/manual/en/latest/render/shader_nodes/shader/diffuse.html?highlight=diffuse%20bsdf, besucht: 15.09.2019.
- [51] BLENDER 2.80 MANUAL, RGB Node,
https://docs.blender.org/manual/en/latest/render/shader_nodes/input/rgb.html?highlight=color%20picker, besucht: 13.09.2019.
- [52] BLENDER 2.80 MANUAL, Emission,
https://docs.blender.org/manual/en/latest/render/shader_nodes/shader/emission.html

highlight=emission%20node, besucht: 15.09.2019.

- [53] BLENDER 2.80 MANUAL, Material Node,
https://docs.blender.org/manual/en/latest/render/shader_nodes/output/material.html?highlight=material%20node, besucht: 15.09.2019.
- [54] WIKIPEDIA, Low Poly, https://en.wikipedia.org/wiki/Low_poly, besucht: 10.08.2019.
- [55] PATHWAYS, 2016, 5 tips to better 3D modeling,
<https://pathwaystrainingandlearning.wordpress.com/2016/05/04/5-tips-to-better-3d-modeling/>, besucht: 30.09.2019.
- [56] NINTENDO, Super Mario 64, <https://www.nintendo.ch/de/Spiele/Nintendo-64/Super-Mario-64-269745.html#Galerie>, besucht: 30.08.2019.
- [57] SUPERHOT VR, Total immersion in SUPERHOT universe,
<https://superhotgame.com/vr/>, besucht: 05.08.2019.
- [58] BLENDER 2.80 MANUAL, Pivot Point,
https://docs.blender.org/manual/en/latest/scene_layout/object/editing/transform/control/pivot_point/index.html, besucht: 02.09.2019.
- [59] A+START, 2019, Link - Low Poly (Evolution of Characters in Video Games) - Episode 7, <https://www.youtube.com/watch?v=jF5a6ltOsK0>, besucht: 05.10.2019.
- [60] A+START, 2017, Super Mario - Low Poly (Evolution of Characters in Games) - Episode 1, https://www.youtube.com/watch?v=A2gXyEyy_2U, besucht: 29.09.2019.

9 Abbildungsverzeichnis

Abbildung 1: Das Blender-Logo [2]	3
Abbildung 2: Spring - Blender Open Movie, eines der Projekte des Blender Animation Studio, welches auf YouTube verfügbar ist [7].....	3
Abbildung 3: Neue kontextbezogene Toolbars [14].....	5
Abbildung 4: Vergleich von Cycles (links) und Eevee (rechts) (eigene Darstellung).....	5
Abbildung 5: User Interface (eigene Darstellung).....	6
Abbildung 6: Viewport Gizmo und Ansichtsoptionen (eigene Darstellung).....	7
Abbildung 7: 3D-Cursor und Pivot Point am Ursprung (eigene Darstellung).....	8
Abbildung 8: Mirror-Modifier (eigene Darstellung).....	8
Abbildung 9: Weight Painting (eigene Darstellung)	8
Abbildung 10: Pose des Rig und des gesamten Modells nach Parenting und Weight Painting (eigene Darstellung)	9
Abbildung 11: Objekt, links ohne und rechts mit eigenem Material (eigene Darstellung)	9
Abbildung 12: Rendering-Stages (eigene Darstellung)	10
Abbildung 13: Ein Vertex-Shader ([23]; eigene Darstellung).....	11
Abbildung 14: Ein Fragment-Shader ([23]; eigene Darstellung)	11
Abbildung 15: Berechnung des Diffuse Lighting ([32]; eigene Darstellung)	12
Abbildung 16: Beispiel von positivem und negativem Skalarprodukt ([34]; eigene Darstellung)	13
Abbildung 17: Berechnung der Specular Reflection ([32]; eigene Darstellung).....	14
Abbildung 18: The Legend of Zelda: The Wind Waker, 2002 erschienen [36]	16
Abbildung 19: Cel Shader (links) und Phong Shader (rechts) (eigene Darstellung).....	16
Abbildung 20: Verbundene Nodes (eigene Darstellung)	17
Abbildung 21 Specular BSDF Node (eigene Darstellung).....	17
Abbildung 22: ColorRamp Node (eigene Darstellung)	18
Abbildung 23: Diffuse BSDF Node (eigene Darstellung).....	19

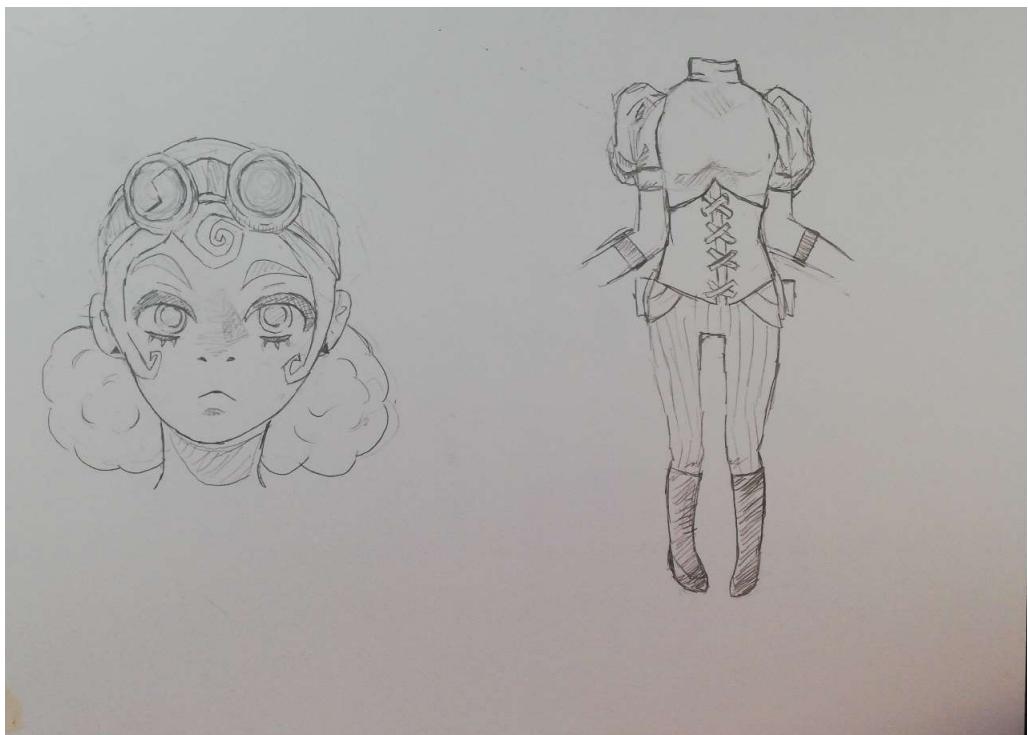
Abbildung 24: RGB Node und Objektfarbe (eigene Darstellung).....	19
Abbildung 25: Leuchteffekt und Emission Node (eigene Darstellung).....	19
Abbildung 26: Low und High Poly (eigene Darstellung)	20
Abbildung 27: Super Mario 64, ein Videospiel, welches 1996 erschien, gilt heutzutage als Low Poly [56]	20
Abbildung 28: SUPERHOT VR, ein Videospiel, welches 2016 erschienen ist [57]	21
Abbildung 29: Entwurf des in Blender zu erstellenden Charakters (eigene Darstellung).....	22
Abbildung 30: Eingefügte Vorlagen im 3D-Viewport (eigene Darstellung)	23
Abbildung 31: Zylinder (eigene Darstellung).....	24
Abbildung 32: Gespiegelte Zylinder (eigene Darstellung)	24
Abbildung 33: Erstellung der Beine (eigene Darstellung).....	24
Abbildung 34: Oberkörper und Arme (eigene Darstellung)	24
Abbildung 35: Erstellungsprozess des Kopfs (eigene Darstellung)	24
Abbildung 36: Erstellungsprozess der Hände (eigene Darstellung).....	25
Abbildung 37: Loop-Cut (eigene Darstellung).....	25
Abbildung 38: Details am Kopf des Modells (eigene Darstellung)	25
Abbildung 39: Leuchtende Materials (eigene Darstellung).....	26
Abbildung 41: Weight Painting des Daumens (eigene Darstellung)	26
Abbildung 40: Mit Hilfe des Rig bewegtes Bein (eigene Darstellung).....	26
Abbildung 42: Cel Shader in Blender, Node Group (eigene Darstellung).....	27
Abbildung 43: Inhalt der Node Group, Cel Shader in Blender (eigene Darstellung)	28
Abbildung 45: Render des Modells in Renn-Pose (eigene Darstellung).....	31
Abbildung 44: Render des Modells in T-Pose (eigene Darstellung)	31
Abbildung 46: Zweiter Render des Modells in rennender Pose (eigene Darstellung)	31
Abbildung 47: Render des Modells in ängstlicher Pose (eigene Darstellung)	32
Abbildung 48: Render des Modells in verlegener Pose (eigene Darstellung)	33

Anhang

Charakterentwurfsskizzen

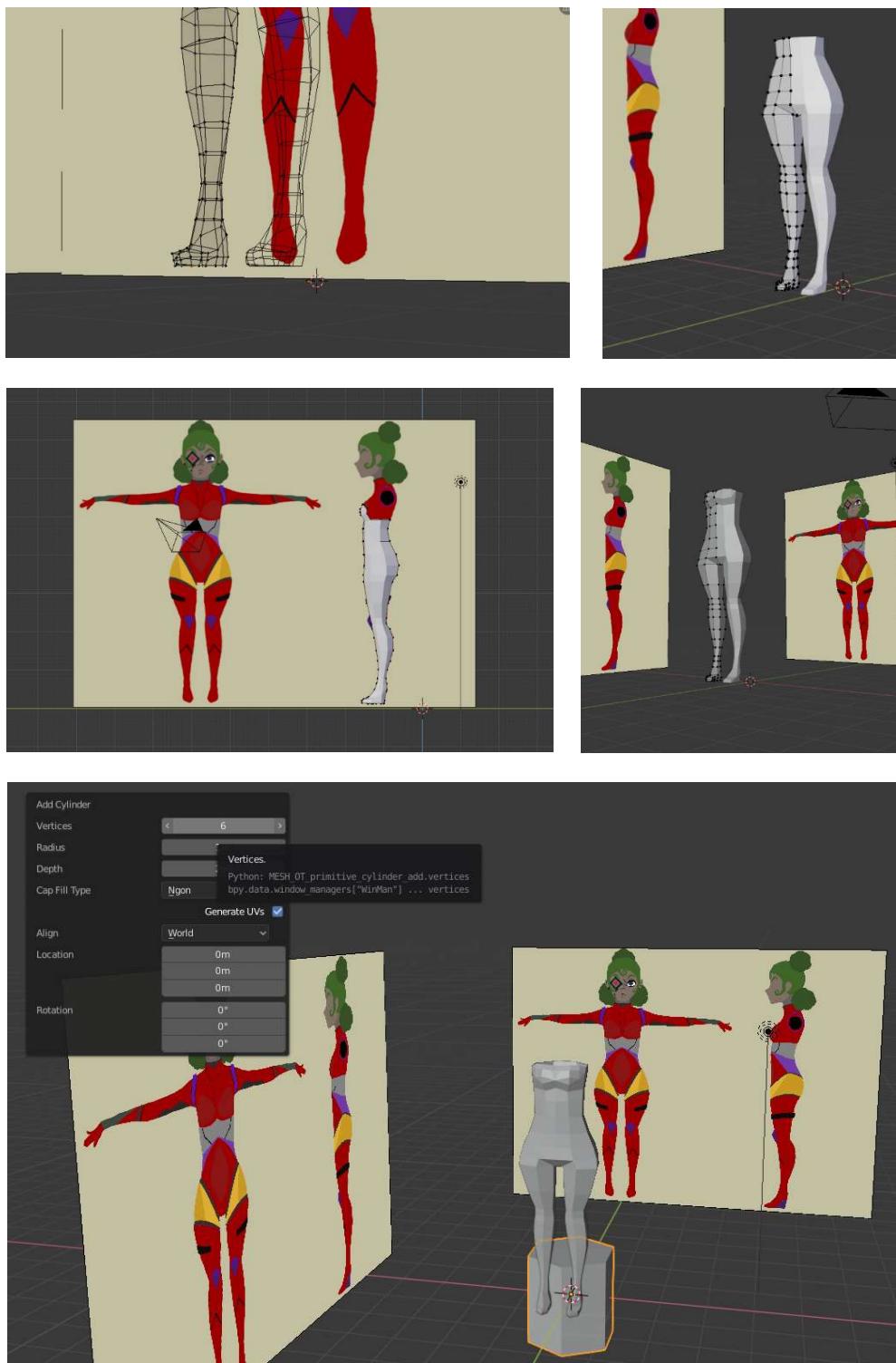


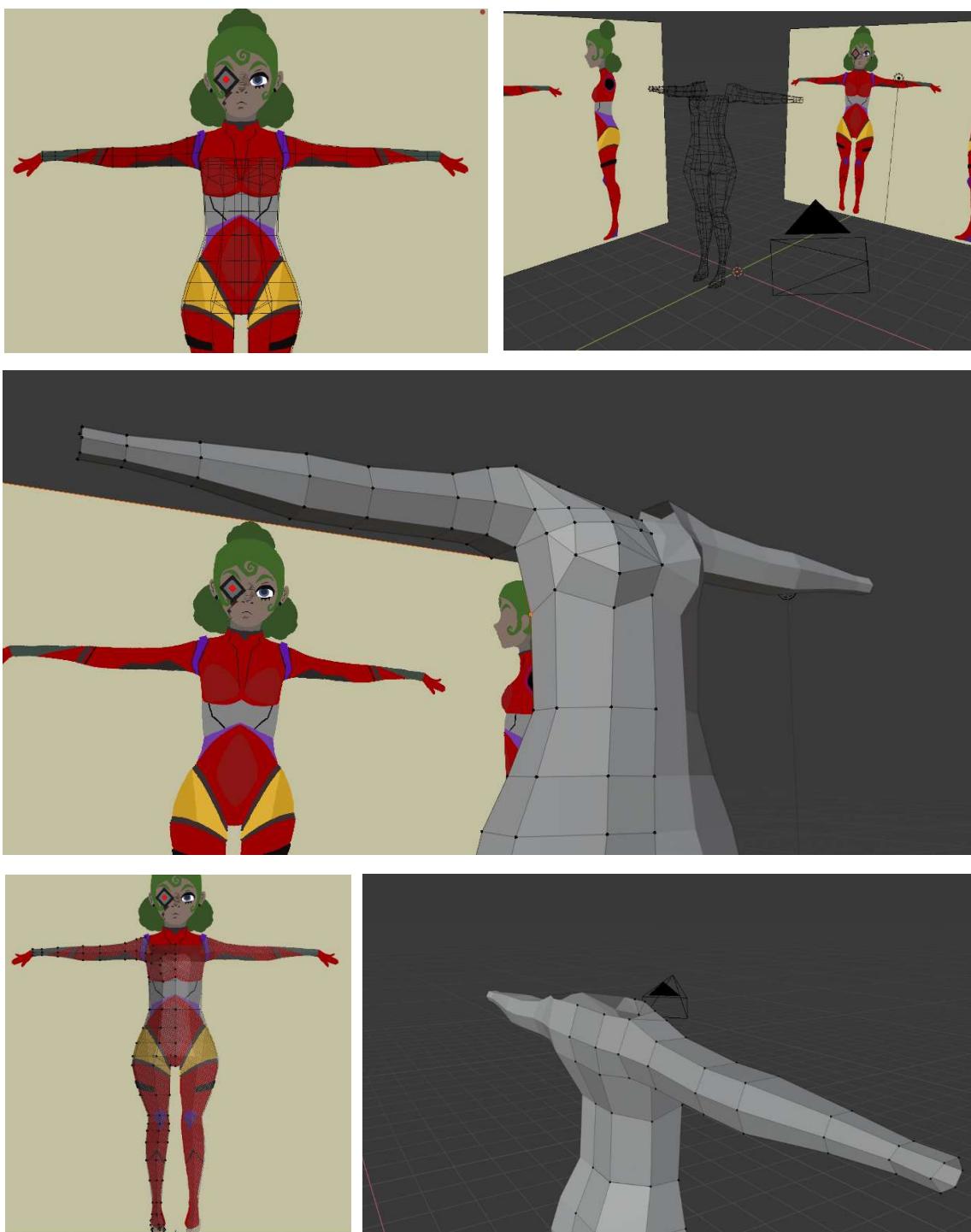


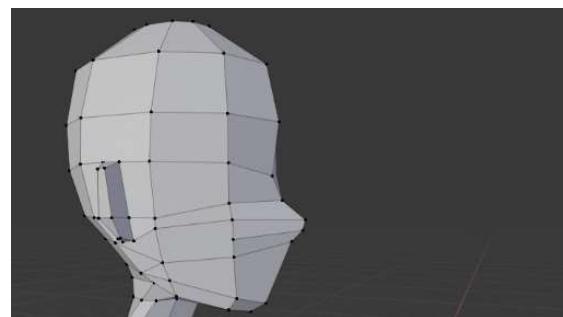
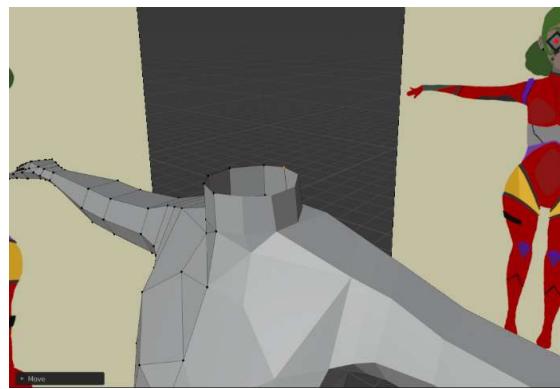
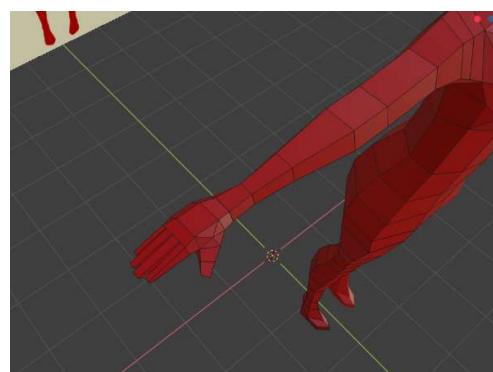
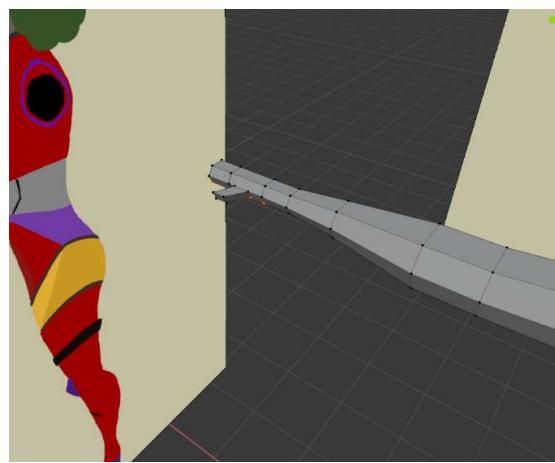
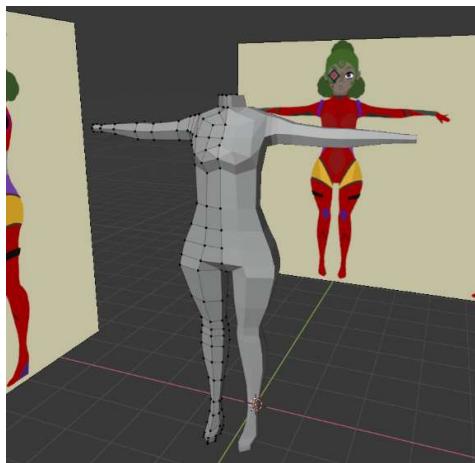


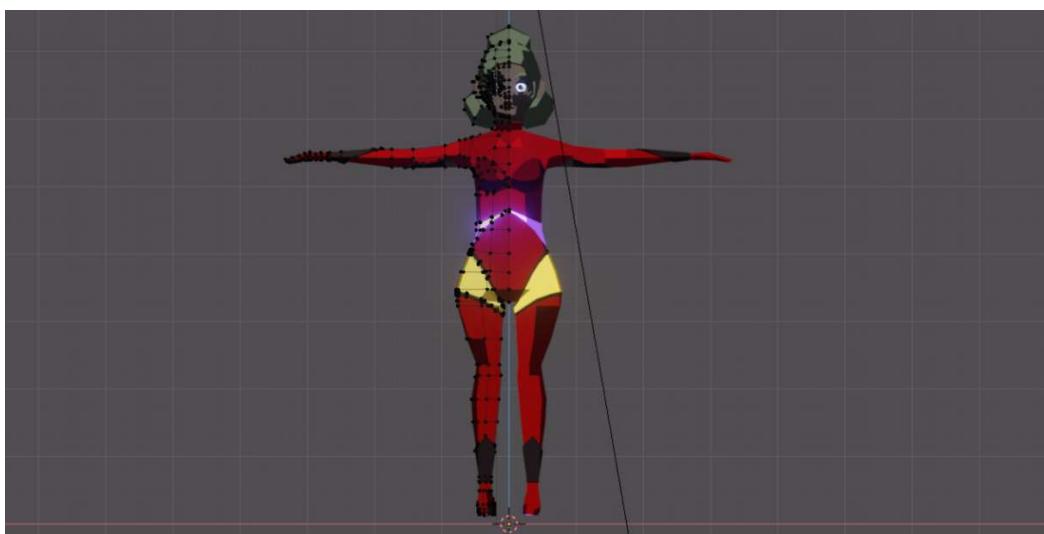
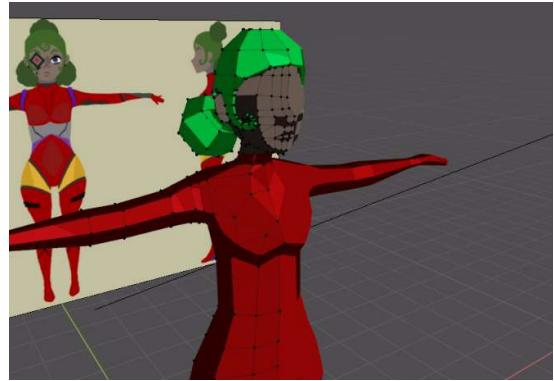


Screenshots des Erstellungsprozesses









Blender 2.8 Hotkey-Sheet

Hotkey	Funktion
F3	Suche nach Funktion
X	Löschen
G (und Koordinatenachse)	Bewegen (auf Koordinatenachse)
S (und Koordinatenachse)	Skalieren (auf Koordinatenachse)
R (und Koordinatenachse)	Rotieren (auf Koordinatenachse)
Ctrl – Tab	Mode wechseln
NumPad 1 oder 3 oder 7 oder 0 oder 5	Ansicht: Front, Rechts, Top, Kamera, Orthogonal
Ctrl – NumPad	Gegenansicht
Shift – A	Neues Objekt hinzufügen
1 oder 2 oder 3	In Edit Mode: Vertex-, Edge-, Face-Auswahl
Mittlere Maustaste	Perspektive ändern
Mittlere Maustaste – Ctrl	Zoomen
Mittlere Maustaste – Shift	In der Szene bewegen
Shift – S	3D-Cursor platzieren
A	Alles markieren
B	Box-Auswahl
Ctrl – rechte Maustaste	Lasso-Auswahl
Shift – linke Maustaste	Mehrere Objekte auswählen und Auswahl aufheben
Alt – R	Rotation zurücksetzen
Alt – G	Position zurücksetzen
Alt – S	Skalierung zurücksetzen
Alt – M	Verschmelzen
Ctrl – R	Loop Cut
K – linke Maustaste – Enter	Knife-Tool
F	Faces aus gewählten Vertices erstellen
Ctrl – P	Set Parent
(Im Weight Paint Mode) rechte Maustaste	Weight Paint-Brush Menü
F12	Render des Bildes

*die Hotkeys beziehen sich auf die Windows-Version von Blender 2.8

Redlichkeitserklärung

Wir erklären hiermit, dass

- diese Arbeit weder ganz noch teilweise abgeschrieben noch kopiert oder aus dem Internet übernommen wurde,
- der Quellennachweis gemäss den Vorgaben des Handbuches Projekts korrekt und vollständig ist,
- die dargestellten Daten und Resultate von den Unterzeichnenden selbst und gemäss den Vorgaben des Handbuches Projekte erhoben und verarbeitet wurden. DFD

Ort	Datum	Name	Unterschrift
Aarau	16.10.2019	Cansu Gül	<i>C. gül</i>
Aarau	16.10.2019	Layla Hamad	<i>Layla Hamad</i>