Ziel der vorliegenden Bachelorarbeit ist es, zu erläutern wie 3D-Echtzeitrendering funktioniert und wie dies mit der Software Ventuz Designer Version 4 von der Firma Ventuz Technology AG durchgeführt wird. Sie ermöglicht dem Anwender, ohne Programmierkenntnis, auf die Funktionen der Grafikkarte über DirectX zuzugreifen. Es wird Grundwissen zum Thema computergestützte Bildgenerierung vermittelt, um ein besseres Verständnis für die dahinter liegenden Prozesse und Verfahren zu entwickeln.
Die Firma Ventuz Technology AG wurde 2004 von verschiedenen Personen aus den Bereichen Fernseh, Veranstaltung und Postproduktion gegründet. Sie bietet hardware-offene Software-Lösungen für die Erstellung und das Ausspielen von echtzeitfähigen, interaktiven, datengesteuerten Grafiken an. Die angebotene Anwendung Ventuz Director dient zur Steuerung der im Ventuz Designer erstellten Inhalte, welche von Unternehmen wie ARD, Porsche, Microsoft und Samsung verwendet werden [19].

Bernburg
Dessau
Köthen

Hochschule Anhalt
Anhalt University of Applied Sciences

emw
Fachbereich
Elektrotechnik, Maschinenbau
und Wirtschaftsingenieurwesen

Bachelorarbeit

zur Erlangung des akademischen Grades
Bachelor of Engineering (B. Eng.)

Robert Altmann

Vorname Nachname

Medientechnik, 2009, 4051429

Studiengang, Matrikel, Matrikelnummer

Thema:
3D-Echtzeitrendering mit Ventuz

Prof. Dr. Steffen Strauß

Vorsitzende(r) der Bachelorprüfungskommission/1. Prüfer(in)

Prof. Dr. Matthias Schnöll

2. Prüfer(in)

08. 01. 2016

Abgabe am

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ARD Arbeitsgemeinschaft der öffentlich-rechtlichen Rundfunkanstalten

FLOPS Floating Point Operation Per Second → Fließkommaoperationen pro Sekunde

IC integrated circuit → integrierter Schaltkreis

DAE Digital Asset Exchange → digitaler Anlangen Austausch

CAM Computer Aided Manufacturing → computergestützte Herstellung

CAD Computer Aided Design → computergestütztes Design

CPU Central Processing Unit → zentrale Recheneinheit

APU Accelerated Processing Unit → beschleunigte Verarbeitungseinheit

NURBS non-uniform rational B-Splines

$Node$ Kunstwort von Ventuz für einen Baustein, abgeleitet aus dem Englischen für Knoten

KAPITEL 1

Einführung

In diesem einführenden Kapitel wird ein kurzer Abriss der Entwicklung der Computertechnik dargestellt. Ein Computer[1] ist ein Gerät, voll programmierbar mit Hilfe von Rechenvorschriften, das Daten verarbeitet. Die ersten Geräte wurden in den 1940er Jahren von Konrad Zuse, John Prespar Eckert und John William Manchly entwickelt. Damals wurden die elektronischen Rechenwerke Großrechner genannt und beschränkten sich allein auf die Verarbeitung von Zahlen [5, S. 4 f.].
Der Zuse Z22 benötigte eine Stellfläche von mindestens 16 m^2, hatte eine Leistungsaufnahme von ca. 3.500 Watt, wog ca. 1.300 kg und der Prozessor war mit Elektronenröhren gebaut. Er hatte eine mittlere Rechengeschwindigkeit von 25 FLOPS und benötigte für das Ziehen einer Quadratwurzel 150 ms [16]. Zum Vergleich bietet eine moderne Grafikkarte, wie die Quadro M4000 vom Hersteller Nvidia, eine Rechengeschwindigkeit von 2,5*10^{12} FLOPS bei einer Leistungsaufnahme von 120 Watt [17].
Wichtige Meilensteine für diese rasante Entwicklung waren der Aufbau von Prozessoren mittels Transistoren in den 60er Jahren, die Integration von kompletten Schaltkreisen in ein Chipgehäuse, (IC-Technik genannt) und die anhaltende Miniaturisierung dieser Bauelemente. Daher stammt der Begriff Mikroprozessor [5, S. 88].
In der Abbildung 1.1. ist der Trend der permanent steigenden Rechenleistung über die letzten Jahrzehnte veranschaulicht.

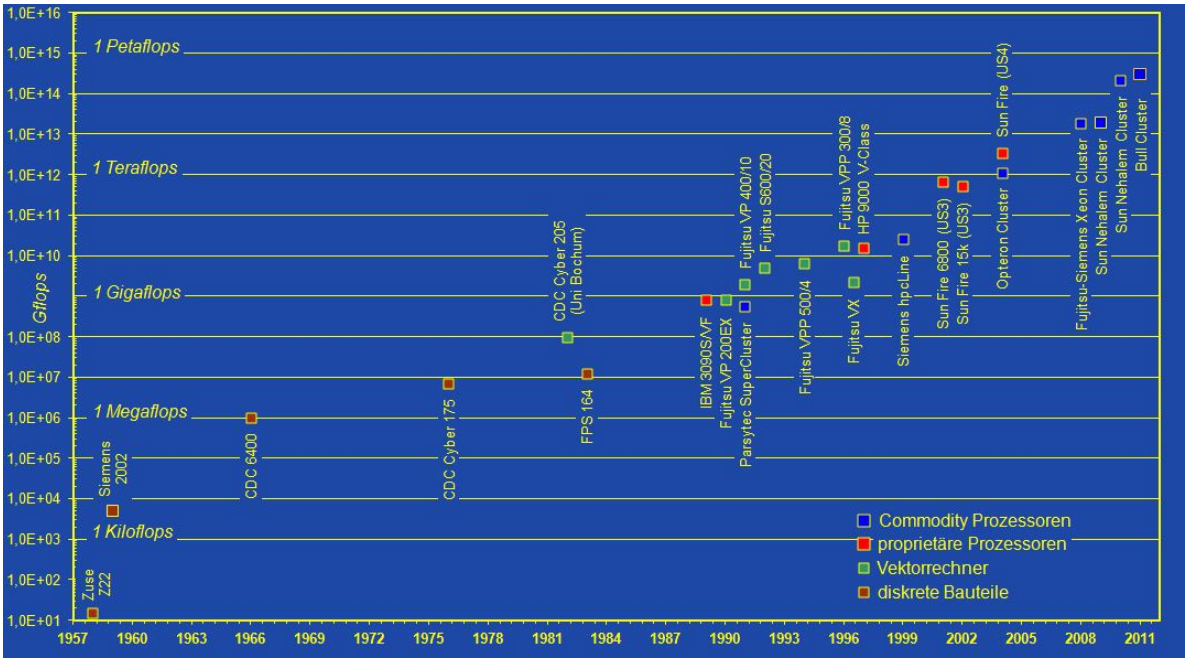


Abbildung 1.1.: Überblick über installierte Großrechner mit FLOPS Angabe in der Rheinisch-Westfälisch Technischen Hochschule Aachen
[11]

[1] lat. computare → berechnen

Das Bestreben die Fertigung von Konstruktionen und ihre technischen Zeichnungen zu automatisieren, verhalf der Computertechnik zum Durchbruch [7, S. 336]. Anfänglich berechnete der Computer die numerischen Werte für die Positionierung des Fräskopfes bei NC-Werkzeugmaschinen, welche per Papier- oder Metall-Lochstreifen vom Großrechner übergeben wurden. Erst ab den 70er Jahren wurde die Berechnung der Positionen in den IC-Chip integriert und durch Interpolation der Fahrten wurden gleichmäßige Bewegungen des Fräskopfes möglich [22, S. 33]. Dies war der Grundstein des heutigen CAD bzw. CAM.
Dabei hat sich der Aufbau eines Computer von 1946 bis heute nicht verändert. Nach wie vor ist die Von-Neumann-Architektur als Referenzmodell gebräuchlich. Sie besteht aus:

- Recheneinheit
- Steuereinheit
- Buseinheit
- Arbeitsspeicher und externe Speicher
- Ein- und Ausgabe

In momentan erhältlichen CPU sind Recheneinheit und Steuereinheit zusammengefasst [5, S. 11].

1.1. Computergrafik

In der klassischen Computergrafik werden aus einer abstrakten Objektbeschreibung ein oder mehrere zeitlich aufeinander folgende Bilder synthetisiert [10, S. 6]. Es besteht die Möglichkeit, die Objekte aus verschiedenen Blickwinkeln zu betrachten. Dadurch entsteht der Vorteil, dass sich ein Betrachter interaktiv durch die 3D-Szene agieren kann. Ein Nachteil dieser Technik ist, dass realitätsgetreue Darstellungen sehr rechenaufwendig sind [10, S. 19].
In den Anfängen wurden die Grafiken mit der CPU berechnet und im Laufe der Jahre immer weiter auf externe Grafikchips ausgelagert. Mitte der 90er Jahre war der Durchbruch der 3D-Beschleunigung. Der Grafikchipsatz Voodoo Graphics vom Hersteller 3dfx war der Meilenstein für den privaten Anwender. Auf dem professionellen Markt waren bereits seit den 80er Jahren Grafikkarten für CAD/CAM Programme erhältlich [14]. Diese Karte wurde zusätzlich neben die 2D-Grafikkarte gesteckt und parallel betrieben. Gegenwärtig finden 2D- und 3D-Berechnung auf der selben Grafikkarte statt, wenn die Grafikkarte die Programmierschnittstelle DirectX oder OpenGL bedient [10, S. 2]. Ein Trend zur Integration des Grafikchips in die CPU, ist derzeit festzustellen. Der neue Begriff APU hat sich für diesen Prozessor etabliert [18]. Zur Darstellung einer 3D-Szene durchlaufen die Daten die sogenannte Renderpipeline. Der Designer von Ventuz ist an die Pipeline von DirectX 9.c gebunden. In der Abbildung 1.2. ist der grobe Aufbau dieser DirectX Pipeline zu sehen. Die Pipeline besteht aus konzeptuellen Stufen. Jede Stufe kann weitere Unterstufen beinhalten und bildet so wieder eine eigenständige Pipeline [1, S. 12 f.].

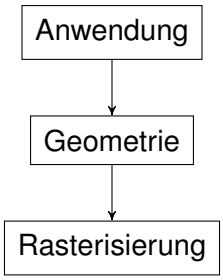


Abbildung 1.2.: Allgemeiner Aufbau einer Pipeline
[1, S. 13 vgl.]

1.2. Echtzeit

Oberste Priorität ist die Generierung eines realitätsnahen Bildes. Um dies zu berechnen, benötigt das System unterschiedliche Zeiten. Diese Zeit ist abhängig von der verwendeten Computerhardware[2]. Um mit dem Ventuz Designer zu arbeiten, ist ein einfaches handelsübliches System ausreichend, solange der Grafikprozessor DirectX 9.c unterstützt.
Allgemein lässt sich die Generierung von Computergrafiken in zwei Kategorien einteilen, in die echtzeitfähigen Systeme und die nicht-echtzeitfähigen Systeme. Die nicht-echtzeitfähigen Systeme benötigen lange Rechenzeit zur Generierung eines Bildes. Solche Bildberechnungen finden vor allem in der Filmproduktion Anwendung. Zum Beispiel wurde für den Animationsfilm Toy Story von Pixar eine große Rechenanlage installiert, die für das Generieren eines Bildes bis zu 15 Stunden benötigte [15].
Zur Unterscheidung dieser Kategorien wurde eine willkürliche Trennlinie gezogen. Bildwiederholraten, die länger als eine Sekunde dauern, gehören zu den nicht-echtzeitfähigen Systemen. Systeme, die schneller als 1 Hz sind, gehören zu den Echtzeit-Renderern. Diese Systeme können interaktive ausgelegt sein. Je nach Anforderung findet eine weitere Einteilung statt. Zum Einen in die sogenannte weiche Echtzeit mit Bildwiederholraten von 1-50 Hz, bei der eine Toleranz zwischen Eingabe und generiertem Bild akzeptiert wird. Typische Vertreter sind Unterhaltungssysteme wie Xbox, Playstation sowie die professionellen CAD-Programme. Zum Anderen die harte Echtzeit mit Bildwiederholraten über 50 Hz, bei dem der Anwender zwischen Eingabe und Ausgabe keine Verzögerung bemerkt [10, S. 25]. Für die Toleranzgrenzen hat man die Analyse von Bewegungsprozessen des menschlichen Auges herangezogen. Die relative Integrationszeit der Photorezeptoren beträgt, abhängig von den Beleuchtungsverhältnissen, etwa 1/50 - 1/10 Sekunde. Um keine separaten Bilder sondern eine einheitliche Bewegung wahrzunehmen, benötigt man Bildwiederholraten über 50 Hz [20, S. 114].
Wie in Abbildung A.1 zusehen ist, lassen sich im Ventuz $ConfigurationEditor$ unter $AVConfiguration$ verschiedene Bildwiederholraten einstellen, um sie an die Anforderungen und Bildwiedergabesysteme anzupassen. Mit dem Configuration Editor ist es außerdem möglich, verschiedene Ein- und Ausgangswege zu konfigurieren.

[2] verwendete Hardware des Systems mit dem diese Arbeit entstanden: AMD Opteron 246 Dual Prozessor, Arbeitsspeicher 4 GByte, Grafikkarte Nvidia Quadro FX3700, Betriebssystem Windows 7 64bit

KAPITEL 2

Modellierung

Bevor eine Ausgabe auf dem Bildschirm stattfindet, muss eine Szene mit 3D-Modellen erstellt werden. Ventuz bietet dem Anwender verschiedene Grundprimitive an, wie auf Abbildung A.2 zu sehen ist. Auf diese Grundprimitive, im Programmcode implizit abgelegt, kann zurückgegriffen werden. Möchte der Anwender andere Modelle verwenden, müssen diese zuerst importiert werden. Danach kann das Modell mit der $MeshLoader\text{-}Node$ in das Projekt bzw. Szene einfügt werden. Der Ventuz Designer unterstützt beim Importieren von Modellen die Formate Collada DAE, Alias/Wavefront, Alias Realtime Graphics und DirectX.
Die Generierung von 3D-Modellen nimmt ein eigenständiges Gebiet in der Computergrafik ein. Je nach Anwendungsfall wurden verschiedene Modelltypen und unterschiedliche Methoden zur Modellierung entwickelt. Es können real existierende oder entworfene Objekte durch einen Anwender manuell, durch ein Programm, durch automatisches Ableiten von Messdaten oder aus Datensätzen modelliert werden. Hierbei bedient man sich einem Teilgebiet der Mathematik, der konstruktiven Geometrie[1] und beschreibt diese Objekte exakt oder näherungsweise [21, S. 9] [10, S. 72] [3, S. 91 f.].

2.1. Modelltypen

Basis eines jeden 3D-Modells ist deren euklidischer Informationsgehalt. Anhand dieser abstrakten Beschreibung lassen sich die Modelle in verschiedene Kategorien einordnen [6, S. 19 ff.] [21, S. 41 ff.]:

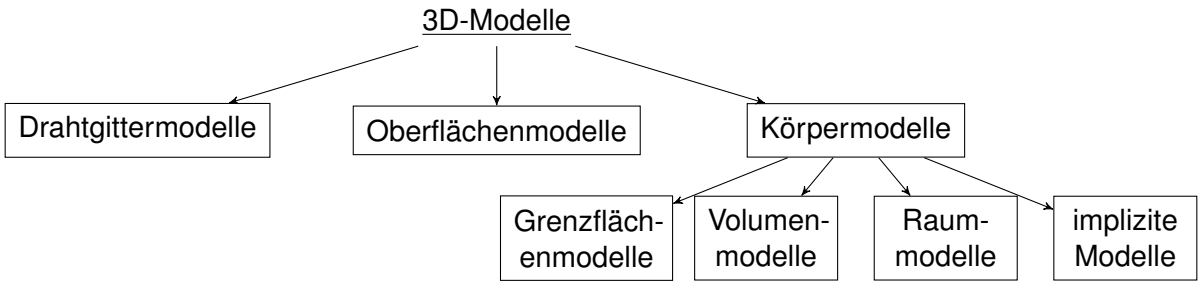


Abbildung 2.1.: Klassifizierung der unterschiedlichen 3D-Modelltypen
[6, S. 19 vgl.]

- **Drahtgittermodelle:** Das Drahtgittermodell ist die einfachste Lösung und beschreibt ein Objekt nur durch Flächenkanten bzw. Geraden. Es enthält die wenigsten Informationen

[1] Geometrie hat seinen Wortursprung in Griechenland und bedeutet Landvermessung. Nach Definition ist die klassisch darstellende Geometrie die Übertragung des dreidimensionalen Anschauungsraum auf die Zeichenebene. Die konstruktive Geometrie bindet analytische Geometrie mit ein, um am Objekt zu operieren [9, S. 11].

und ist in seiner Darstellung mehrdeutig. Durch Bearbeitung kann das Modell vom Anwender in ein Oberflächen- oder Grenzflächenmodell überführt werden. Eine weitere Möglichkeit ist die Umschaltung von Ansichten. Ventuz unterstützt, wie in Abbildung A.3 zu sehen ist, die Umschaltung zur Drahtgitteransicht. Somit kann eine Szene schnell überprüft werden.

- **Oberflächenmodelle:** Dies sind Modelle, die aus einer Vielzahl von Ebenen oder mathematisch gekrümmten polygonalen[2] Flächen bestehen. Diese Flächen können miteinander verknüpft werden und komplexe Formen annehmen. Zum Beispiel bei der Berechnung der Außenhaut von Kraftfahrzeugen, Schiffen und Flugzeugen oder der Darstellung von Schrift. In Abbildung A.4 erkennt man den Unterschied zwischen Drahtgittermodell und gerenderten Endergebnis des Buchstabens. Dabei wird auf verschiedene Verfahren wie Hermite, Bézier, Spline oder NURBS zurückgegriffen, um weiche Oberflächenformen zu erzeugen. Im Beispiel handelt es sich um die Schrift Calibri, welche als quadratische Bézier-Spline gespeichert ist [13]. Umgangssprachlich werden diese Verfahren zum Sammelbegriff Vektorgrafik verallgemeinert.

- **Körpermodelle:** Dies ist der Obergriff für Modelle, die durch eine topologische Orientierung eine eindeutig festgelegte Innen- und Außenseite besitzen. Sie können weiter durch verschiedene Eigenschaften wie Rand-Volumenbezug, Objekt-Raumbezug, implizite oder explizite Beschreibung unterteilt werden. Explizit bedeutet, dass der abzubildende Körper in seiner Datenstruktur vorhanden ist.

- **Grenzflächenmodelle:** Sie bestehen wie die Oberflächenmodelle aus einer unterschiedlichen Anzahl von Polygonen, welche den zu modellierenden Körper approximieren und mit einem Netz umschließen. Der Grad der Genauigkeit wird durch die Anzahl der verwendeten Polygone bzw. Maschen bestimmt. Im Gegensatz zum Oberflächenmodell kann genau unterschieden werden, welcher Punkt außerhalb oder innerhalb des Modells liegt. Die Darstellung dieses Modells ist meist ausreichend, da nur die äußeren Flächen des Körpers sichtbar sind. Wenn das Objekts für mechanische Simulatoren benutzt werden soll, kann das Volumen zusätzlich mit Parametern versehen werden.

- **volumenbezogene Modelle:** Ein typischer Vertreter der volumenbezogenen Modelle ist die Constructive Solid Geometry, wo komplexe Gebilde mit sogenannten Primitiven erzeugt werden. Zu den Primitiven gehören fest definierte Grundkörper wie Quader, Kegel, Kugel, Zylinder, Torus, Pyramide und mehr. Sie werden mit Hilfe von logischen Operatoren wie Vereinigung, Differenz und Schnitt verknüpft. Diese Modelle werden zum Beispiel in CAD Programmen benutzt, um Objekte für Werkzeugmaschinen zu erstellen.

- **raumbezogene Modelle:** Hier wird ein vorhandener Raum in immer kleinere Bereiche unterteilt bis die vorgegebene Minimalgröße erreicht ist. Die elementaren Teile werden Voxel genannt. Das Kunstwort Voxel wurde in die Anlehnung an das Kunstwort Pixel gebildet. Pixel steht für Picture Elements und Voxel für Volume Elements. Voxel können markiert, eingefärbt und mit anderen Eigenschaften versehen werden. Im räumlichen Gitter

[2] Polygon $\rightarrow$ Vieleck

geordnet, können sie kleine Würfel sein. Kleine Kugeln bilden eine Wolke mit Wassertropfen ab. Dieses Modell wird vorrangig in der Medizin bei der Magnet-Resonanz-Tomografie verwendet.

- **implizite Modelle:** Das Objekt wird als reine mathematische Formel angegeben. Zum Beispiel ist $x^2 + y^2 + z^2 = r^2$ die Funktion einer Kugel. Anwendung finden die impliziten Modelle bei Form-verändernden Animationen. Ansonsten sind solche Modelle beschränkt brauchbar, da die Modelldaten durch einen Vorverarbeitungsschritt extrahiert werden müssen und effiziente Algorithmen für Polygonenmodelle den Markt beherrschen.

In der Computergrafik existieren noch weiter Modelltypen. Abbildung A.6 zeigt eine $3D - Text$-Node, mit ihr können 2D-Texte in dreidimensionale umgewandelt werden. Jedoch verlieren diese Schriften ihre Vorteile der Vektorgrafik und die Rundungen werden eckig abgebildet.
Die hier verwendete Auswahl soll sich auf die in der Praxis geläufigsten Modelltypen beschränken [23, S. 45 f.]. Abbildung A.5 zeigt den Import eines Modells mit den verfügbaren Optionen, wie das Modell im Ventuz eigenen Format abgespeichert werden soll. Beim Import kann die Datenstruktur, die Facetten, der lokale Referenzpunkt und die Textur beeinflusst werden. Für ein besseres Verständnis soll nachfolgend auf die Grenzflächenmodelle und ihre Grundlagen näher eingegangen werden.

2.2. Polygon

Alle wichtigen Konzepte vom Grenzflächenmodell lassen sich anhand von Polygonen erklären. Dabei wird die Außenhaut eines zu modellierenden Körpers durch eine Menge markanter Punkte beschrieben. Verbindet man diese Punkte mit einer Geraden untereinander, entsteht ein Polygonzug. Die Geraden zwischen den Eckpunkten heißen Kanten. Ist der erste Punkt mit dem letzten Punkt identisch, dann sprechen wir von einem geschlossenen Polygonzug, wie in Abbildung 2.2 zu sehen ist.

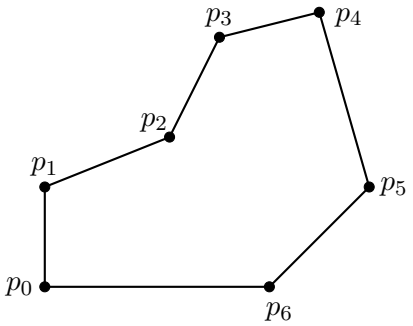


Abbildung 2.2.: Beispiel eines geschlossenen Polygonzugs

Liegen alle Kanten auf einer Ebene, so ist das Polygon planar. Die eingeschlossene Fläche zwischen Kanten eines planaren Polygons heißt Facette[3]. Ein geschlossenes, planares Polygon kann aus mathematischer Sicht als Graph interpretiert und daraus Zusammenhänge zwischen der Anzahl von Ecken, Kanten hergeleitet bzw. überprüft werden. Erfüllt eine Menge von

[3] Facette abgeleitet von englisch Face → Gesicht

geschlossenen, planaren Polygonen gewisse Eigenschaften, so werden sie als Polygonnetz bezeichnet. Zusammenfassend werden aus Polygonen die Grenzflächen des Körpers und aus den Grenzflächen wird das Objekt modelliert [3, S. 191 f.] [23, S. 25 ff.].

2.2.1. Polygonnetz

Man kann die Modellierung mit Polygonnetzen anschaulich erläutern. Der zu beschreibende Körper wird mit dem Netz eingewickelt. Aus diesem Grund wird der Begriff Hüllenbeschreibung oder in Englisch Boundary Representation verwendet. Die häufigsten Vertreter von Polygonnetzen sind die Dreieck- und Viereckknetze. Wobei die momentan erhältliche Hardware für Dreiecksnetze optimiert ist. Viereckknetze werden durch die Triangulation in Dreieckknetze umgewandelt, was die nachfolgenden Berechnungen und die Projektion auf die Projektionsebene erheblich vereinfacht. Abbildung 2.3 zeigt einen Kubus aus einem Viereckknetz. e_n ist die Bezeichnung für die n-Ecke und f_n ist die Bezeichnung für die n-Facette.
Die Weite der Maschen bestimmt den Grad der Genauigkeit. Für gekrümmte Flächen gibt es

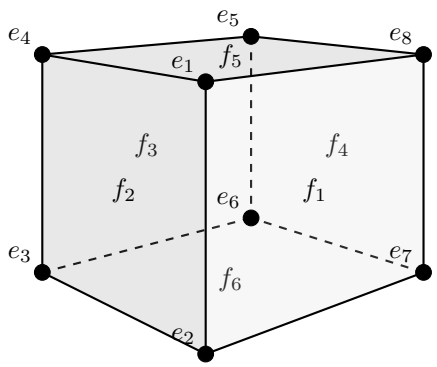


Abbildung 2.3.: Viereckknetz in Form eines Kubus

exaktere Verfahren, bei denen die Krümmung mathematisch beschreiben wird. Eine Möglichkeit ist, durch Polynome die gegebene Kurve zu interpolieren oder sich dieser anzunähern. Erwähnenswert sind die nicht-uniformen rationalen B-Splines kurz NURBS, die eine Verallgemeinerung von nicht-uniformen rationalen B-Splines und nicht rationalen und rationalen Bézierkurven sind. Splines kann man sich anschaulich vorstellen, indem in einen Tisch drei Nägel geschlagen werden und an diesen Nägeln entlang ein Fieberglasstab legt wird. Der Stab versucht die Biegeenergie zu minimieren und oszilliert um die Nägel nicht mehr als nötig. Diesen glatten Kurvenverlauf kann man mit quadratischen Polynomen beschreiben. Der Unterschied zu Bézier-Splines ist, dass die Nägel nicht mehr berührt werden. Es findet nur noch eine Annäherung an den Kontrollpunkt statt [2, S. 341]. Die Lösung bzw. Darstellung eines solchen Verlaufes benötigt sehr viel Rechenzeit und wird auch deshalb in Dreieckknetze umgewandelt.
Es kommt vor, dass auf eine sehr kleine Fläche tausende von Polygonen abgebildet werden müssen. Um die Grafikkarte nicht unnötig zu belasten, wurde ein Optimierungsverfahren entwickelt, das sogenannte Level-of-Detail. Zusätzlich zur Originalgeometrie werden weitere Detailstufen abgespeichert. Aus den unterschiedlichen Objektbeschreibungen lassen sich, je nach Abstand zum Objekt, variable Details abrufen. Ein weiteres Verfahren ist die Tesselation. Das einzelne Polygon wird dabei in weitere Teilpolygone aufgeteilt. Somit erhält das zu modellierende Objekt immer mehr Details.

In der 70er Jahren wurden die Schummerung- und Schattieralgorithmen entwickelt, um den Körper anschaulicher darzustellen. Diese werden im englischen Sprachgebrauch Shader genannt. Dabei wird zu jeder Facette ein Normalvektor, der orthogonal auf der Ebene steht, berechnet. Mit Hilfe des Normalvektors wird der Facette im Laufe des Programms ein Farbwert zugeordnet. Zusätzlich kann zu jeder Ecke ein Ecknormalvektor berechnet werden, um die Möglichkeit zu haben, zwischen zwei angrenzenden Facetten die Farbe zu interpolieren. Die anfallenden Daten werden in einer für sie optimierten Struktur gespeichert. Die Schattierprogramme werden in einem separaten Kapitel ausführlicher beschrieben und haben den Dreiecknetzen bis heute ihre Beständigkeit gesichert.

2.3. Datenstruktur

Es gibt viele Möglichkeiten Modelle zu speichern. Die naheliegende Methode wäre, die Position jeder einzelnen Facette durch Aufzählung ihrer Eckpunkte zu beschreiben. Jedoch birgt dies den Nachteil, dass Speicherplatz benötigt wird. Ein anderer Ansatz beschreibt einzelne Elemente und deren Nachbarschaftsbeziehungen zwischen den jeweiligen Facetten, Kanten und Eckpunkten. Die Eckpunkte werden auch als Knoten bezeichnet. Die Topologie, welche die räumliche Anordnung zueinander beschreibt und die Geometrie werden unabhängig gespeichert. Diese Art und Weise der Ordnung nennt man Datenstruktur. Zum besseren Verständnis werden zwei Strukturen vorgestellt, die sich auf den Kubus in Abbildung 2.3 beziehen [3, S. 197 f.]:

- **Knotenorientierte Struktur:** In einer knotenorientierten Struktur werden alle Knoten explizit abgelegt. Die einzelne Facette wird durch ihre zugehörige Knotenliste definiert und der Knoten durch seine Koordinaten. In Tabelle 2.1 ist die knotenorientierte Struktur zu sehen. Durch ihre geringen Angaben benötigt sie wenig Speicherplatz. Eine Suche nach Kanten und deren zugehörigen Facetten ist jedoch nicht möglich.

Knoten	Koordinaten		Facette	Knoten
e_1	$x_1\ y_1\ z_1$		f_1	$e_8\ e_7\ e_2\ e_1$
e_2	$x_2\ y_2\ z_2$		f_2	$e_1\ e_2\ e_3\ e_4$
e_3	$x_3\ y_3\ z_3$		f_3	$e_6\ e_5\ e_4\ e_3$
e_4	$x_4\ y_4\ z_4$		f_4	$e_5\ e_6\ e_7\ e_8$
e_5	$x_5\ y_5\ z_5$		f_5	$e_8\ e_5\ e_4\ e_1$
e_6	$x_6\ y_6\ z_6$		f_6	$e_7\ e_6\ e_3\ e_2$
e_7	$x_7\ y_7\ z_7$			
e_8	$x_8\ y_8\ z_8$			

Tabelle 2.1.: Knotenorientierte Datenstruktur

- **Kantenorientierte Struktur:** In einer kantenorientierten Struktur werden, wie in der knotenorientierten Struktur, ebenfalls alle Knoten explizit mit Koordinaten abgelegt. Zusätzlich werden die Kanten aus einer Liste von Knoten abgespeichert. Die einzelne Facette wird durch ihre zugehörige Kantenliste definiert. In Tabelle 2.1 ist die kantenorientierte

Struktur zu sehen. Alle Objekt relevanten Merkmale sind in dieser Struktur enthalten.

Kante	Knoten		Knoten	Koordinaten		Facette	Kanten
k_1	$e_1\ e_2$		e_1	$x_1\ y_1\ z_1$		f_1	$k_{10}\ k_9\ k_7\ k_1$
k_2	$e_2\ e_3$		e_2	$x_2\ y_2\ z_2$		f_2	$k_4\ k_3\ k_2\ k_1$
k_3	$e_3\ e_4$		e_3	$x_3\ y_3\ z_3$		f_3	$k_{12}\ k_{11}\ k_5\ k_3$
k_4	$e_4\ e_1$		e_4	$x_4\ y_4\ z_4$		f_4	$k_8\ k_7\ k_6\ k_5$
k_5	$e_5\ e_6$		e_5	$x_5\ y_5\ z_5$		f_5	$k_{12}\ k_9\ k_8\ k_4$
k_6	$e_6\ e_7$		e_6	$x_6\ y_6\ z_6$		f_6	$k_{11}\ k_{10}\ k_6\ k_2$
k_7	$e_7\ e_8$		e_7	$x_7\ y_7\ z_7$			
k_8	$e_8\ e_5$		e_8	$x_8\ y_8\ z_8$			
k_9	$e_1\ e_8$						
k_{10}	$e_2\ e_7$						
k_{11}	$e_3\ e_6$						
k_{12}	$e_4\ e_5$						

Tabelle 2.2.: Kantenorientierte Datenstruktur

Je nach verwendetem Modellierer werden verschiedene Datenstrukturen und Kriterien für die Nummerierung von Knoten, Kanten und Facetten verwendet. Allgemein ist zu beachten, dass beim Import von systemfremden Datenstrukturen, Komplikationen bei der Darstellung des Objektes auftreten können [21, S. 66].

KAPITEL 3

Dreidimensionaler Raum

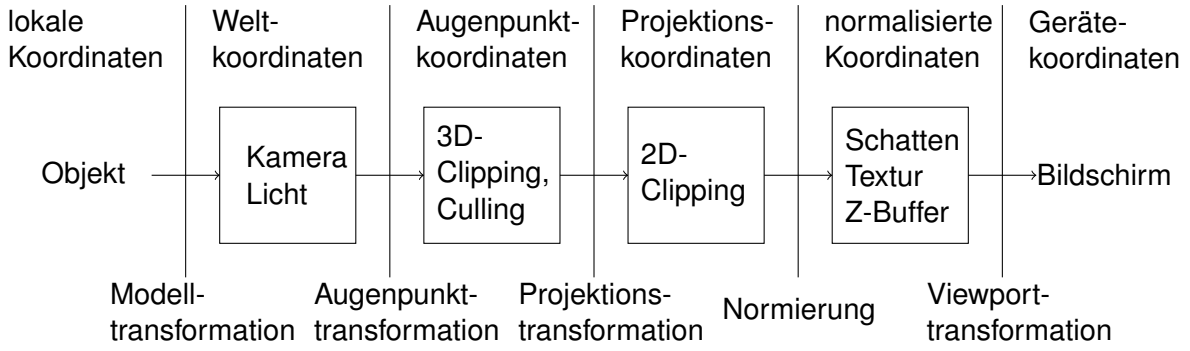


Abbildung 3.1.: Allgemeiner Aufbau der Geometriestufe
[10, S. 123 vgl.]

Objekte durchlaufen von der abstrakten Beschreibung in der 3D-Szene bis zu deren zweidimensionalen Abbildung auf einem Ausgabegerät mehrere Prozesse [3, S. 72]. Diese Prozesse sind in der Form einer Pipeline aufgebaut und können grob klassifiziert werden, wie in Abbildung 3.1 zu sehen ist. Abbildung 3.2 zeigt den kompletten Prozess für die Zentralprojektion. Im Kapitel 1.1 wurde eine allgemeine Form der Renderpipeline von DirectX vorgestellt. Im nachfolgenden Kapitel werden die einzelnen Räume der Geometriestufe und ihre Operationen, Transformationen näher erläutert. In Ventuz können die einzelnen Parameter der Prozesse beeinflusst werden. In der $Toolbox$, wie in Abbildung A.7 dargestellt, stehen verschiedene $Nodes$ zur Verfügung. Wobei alle Prozesse nach der Projektionstransformation sich in der Stufe der Rasterisierung befinden. Schattier- und Texturoperationen werden in separaten Kapiteln betrachtet.

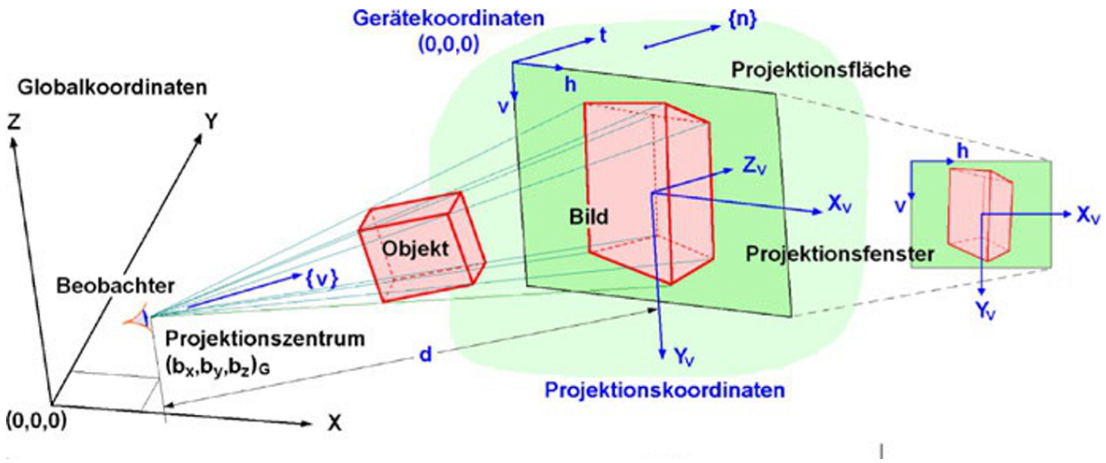


Abbildung 3.2.: Veranschaulichung der Zentralprojektion im 3D-Raum
[21, S. 101]

3.1. Koordinatensysteme

Unsere physikalische Erfahrungswelt und die enthaltenen Objekte können abstrakt in geometrischen Punkträumen dargestellt werden. Ein zentraler Begriff in der linearen Algebra ist der Vektorraum. Der Vektorraum macht jedoch keine Unterschiede zwischen einem Punkt oder einem Vektor. Aus diesem Grund erweitert man den Vektorraum zu einem affinen Raum bzw. euklidischer Punktraum [3, S. 10]. Ein dreidimensionaler euklidischer Punktraum wird durch paarweise sich rechtwinklig schneidende Koordinatenachsen mit einem gemeinsamen Nullpunkt aufgespannt, wie in Abbildung 3.3 zu sehen ist.

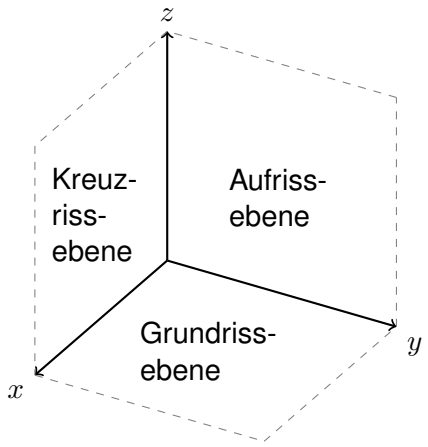


Abbildung 3.3.: Kartesisches Koordinatensystem

Die Koordinatenachse gleicht einer orientierten Zahlengerade mit vielfachen Einheitspunkten im definierten Intervall vom Nullpunkt aus. Ein Koordinatensystem ist rechts-orientiert, wenn die orientierten Zahlengraden um eine positive Vierteldrehung (90° gegen den Uhrzeiger) orthogonal zueinander stehen. Punkte werden in diesem Orientierungssystem als Zahlentripel dargestellt [4, S. 25 f.].
Die Orthonormalbasis x1 spannt sich nach links auf und wird als x-Achse bzw. Abzisse bezeichnet. Die Orthonormalbasis x2 spannt sich nach rechts auf und wird als y-Achse bzw. Ordinate bezeichnet. Die Orthonormalbasis x3 spannt sich nach oben auf und wird als z-Achse bzw. Kote bezeichnet. Die Ebene zwischen x-y heißt Grundrissebene. Die Ebene zwischen y-z heißt Aufrissebene. Die Ebene zwischen x-z heißt Kreuzrissebene [9, S. 30].

3.1.1. Weltkoordinaten

Alle 3D-Objekte werden von ihren lokalen Koordinaten in das Weltkoordinatensystem überführt. Diesen Vorgang bezeichnet man Modelltransformation. Somit sind alle Koordinaten für die gesamte Szene verfügbar. Das Weltkoordinatensystem spannt den globalen Raum auf. In diesem System werden die räumlichen Beziehungen der einzelnen Objekte zueinander definiert. Handelt es sich um bewegte Objekte, werden diese von einem Animationssystem durch eine zeitabhängige Transformation von Bild zu Bild überführt.
Lichtquellen werden ebenfalls im globalen Raum festgelegt, die die Szene beleuchten. Wenn

die Schattierfunktionen verwendet werden, ist die Berechnung der Normalen, Festlegung von Oberflächeneigenschaften für Textur und Farbe die letzte Transformation in diesem System [10, S. 121] [6, S. 72 f.].

3.1.2. Augenpunktkoordinaten

Analog zur Modelltransformation gibt es die Überführung vom Weltkoordinatensystem zum Augenkoordinatensystem, diese nennt man Ansichtstransformation. Hierbei kann der Anwender einen Beobachterstandpunkt im Koordinatensystem, aus der auf die Szene geschaut wird, festlegen. Vorteilhaft ist hier, wenn die Blickrichtung mit der z-Achse übereinstimmt. Ebenso werden in dieser Stufe die Projektionsart und Sichtvolumen festgelegt. In einigen Rendern werden die Modell- und Ansichtstransformation als ModelView-Matrix zusammengefasst [10, S. 121] [6, S.72 f.].

3.1.3. Projektionskoordinaten

Dies sind Koordinaten, die nach der perspektivischen oder parallelen Projektion auf der Projektionsebene entstehen. Ebenso wurden die Polygone außerhalb des Sichtvolumens und verdeckte Rückseiten entfernt. Diese Verfahren nennt man 3D-Clipping und 3D-Culling. Wenn die z-Achse als Blickrichtung gewählt wurde, spannen die x-y-Achsen die Projektionsebene auf und erleichtern die Überführung auf die Gerätekoordinaten [10, S. 122] [6, S. 72 f.].

3.1.4. Normalisierte Koordinaten

Mit der Normierung werden die Projektionskoordinaten auf einem Einheitsquadrat abgebildet. Dieser Vorgang transformiert die x,y,z-Koordinaten in das Intervall von [-w;+w]. Anschließend findet eine Division durch die Streckungsinverse statt. Dieses Quadrat ist die Bildebene und trägt die Achsenbezeichnung u für die x-Achse und v für die y-Achse. Das z-Buffer Verfahren, welches die Sichtbarkeit von Objekten bestimmt, wird ebenfalls in dieser Stufe angewendet [21, S. 80] [10, S. 122] [6, S. 72 f.].

3.1.5. Gerätekoordinaten

Zuletzt werden die normierten Koordinaten mit der Viewporttransformation auf die Gerätekoordinaten überführt. Das projizierte Bild wird gerastert und de facto die einzelnen Pixel des Bildwiedergabesystems beschrieben [6, S. 72 f.].
Über das $ConfigurationTool$ kann auf die Viewporttransformation geändert werden. Abbildung A.8 zeigt auf der linken Seite die Einstellmöglichkeiten für das Bildwiedergabesystem und auf der rechten Seite könnte zusätzlich das projizierte Bild verzerrt werden.

3.2. Perspektive und Projektion

Die Abbildung eines 3D-Raumes auf einer Ebene heißt Projektion. Ventuz bietet verschiedene Möglichkeiten an, eine Szene abzubilden. Die Ebene wird Projektionsebene genannt und ist

beliebig im Raum platziert. Ebenfalls muss ein Projektionszentrum, das nicht auf der Projektionsebene liegt, festgelegt werden. Eine Gerade vom Projektionszentrum zu einem beliebigen Punkt im Raum heißt Projektionsgerade. Schneidet sich Projektionsgerade und Projektionsebene spricht man von einem Bildpunkt bzw. Riss. Befindet sich das Projektionszentrum endlich entfernt und alle Projektionsgeraden laufen durch einen Punkt, spricht man von Zentralprojektion. Befindet sich das Projektionszentrum unendlich entfernt und alle Projektionsgeraden laufen parallel zueinander, spricht man von Parallelprojektion. In Abbildung 3.4 sind auf der linken Seite drei Parallelprojektionen zu sehen. Befindet sich das Projektionszentrum unendlich entfernt und sind alle Projektionsgeraden orthogonal zur Projektionsebene, spricht man von Normalprojektion. Die Normalprojektion ist ein Spezialfall von der Parallelprojektion. In Abbildung 3.4 sind auf der rechten Seite drei Normalprojektionen abgebildet [9, S. 38 f.] [8, S. 42].

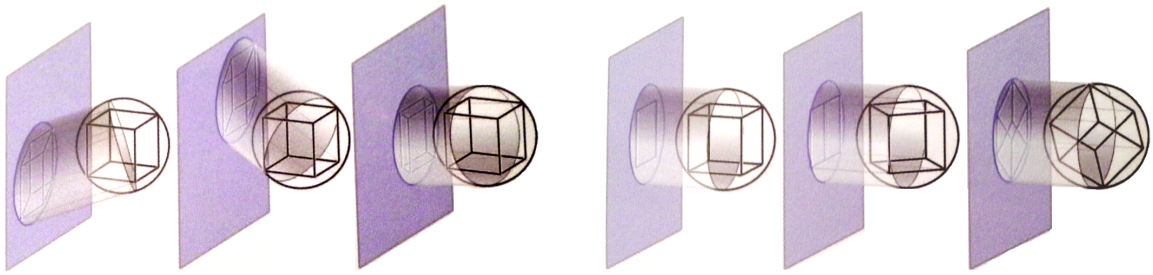


Abbildung 3.4.: Parallelprojektion und Normalprojektion
[8, S. 46]

Im Ventuz Designer wird mit der Zentralprojektion als Standard gearbeitet, da sie dem menschlichen Sehen am ähnlichsten ist. Mit der $Projection\text{-}Node$ und der $PixelSpace\text{-}Node$ kann die Projektionsart und Perspektive geändert werden. Zur Vollständigkeit werden die weiteren Perspektiven kurz erläutert.
Perspektive beschreibt die Parameter, welche auf die Projektion angewendet werden. Für die Zentralprojektion gibt es drei unterschiedliche Perspektiven, die Zentral- , die Vogel- und die Froschperspektive. Bei der Zentralperspektive befindet sich der Beobachterstandpunkt auf einer Höhe mit dem Mittelpunkt einer Szene. Bei der Froschperspektive liegt der Beobachterstandpunkt unterhalb des Mittelpunktes einer Szene. Bei der Vogelperspektive liegt der Beobachterstandpunkt oberhalb des Mittelpunktes einer Szene.
Für die technische Abbildung von Objekten haben sich normierte Perspektiven durchgesetzt. Diese Perspektiven werden als Axonometrie bezeichnet. Dabei können Koordinatenachsen aus ihrer Orthogonalbasis verschoben sein und die entsprechenden Objektkanten skaliert [4, S. 90 f.] [9, S. 63 ff.]. In Abbildung 3.5 sind ein paar Beispiele zu sehen und in Tabelle 3.1 sind die dazugehörigen Parameter und Namen.

Perspektive	α_1	α_2	x	y	z
Isometrie	30°	30°	1	1	1
Dimetrie	42°	7°	$\frac{1}{2}$	1	1
Kabinett	45°	0°	$\frac{1}{2}$	1	1
Kavalier	0°<α_1<90°	α_1 + α_2 = 90°	1	1	1

Tabelle 3.1.: Verschiedene normierte Perspektiven

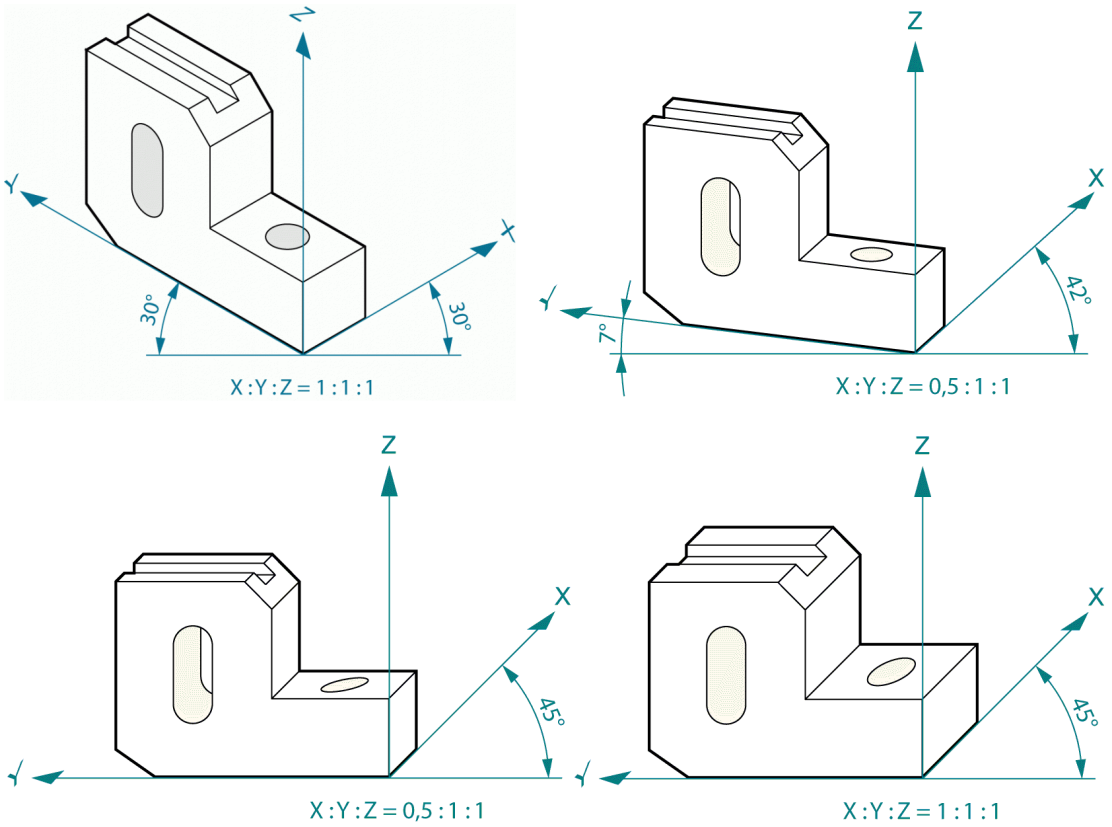


Abbildung 3.5.: Beispiele für verschiedene Perspektiven
[12]

3.3. Homogene Koordinate

Es gibt viele Interpretationsmöglichkeiten der homogenen Koordinate. Eine davon soll nachfolgend vorgestellt werden. Der Grundgedanke ist, dass alle Transformationen auf eine Matrizenmultiplikation zurückgeführt werden. Die Vektormultiplikation wird zum Beispiel mit einer $3 * 3$ Matrix dargestellt. Diese Berechnung ist bei der Translation nicht möglich. Hier wird ein Punkt durch Addition mit einem Vektor verschoben. Um jetzt die Addition als Multiplikation abzubilden, erweitert man Punkte und Vektoren um eine Dimension namens w und überführt alles in eine $4 * 4$ Matrix. Diese Dimension w heißt inverser Streckungsfaktor und ist außer in der Zentralprojektion $w = 1$. $w = 0$ ist unzulässig, da im Renderprozess der Normierung durch diesen Faktor geteilt werden muss [10, S. 124 f.] [21, S. 84 f.].

3.4. Transformationen

Die Grundidee geht von einem fixen Weltkoordinatensystem aus. Um Objekte im Weltkoordinatensystem zu positionieren, müssen Operationen angewendet werden. Mit der $Axis\text{-}Node$ aus der $Toolbox$ können diese manipuliert werden. Diese Operationen sind die Translation, die Skalierung und die Rotation und werden am Objekt im lokalen Koordinatensystem ausgeführt. In Abbildung A.9 kann man dies sehen. Wir betrachten ausschließlich Polygonnetze, die aus einer Menge von Punkten bestehen. Diese können in einer Matrix dargestellt werden. Die Menge an Transformationen lässt sich in einer kombinierten Matrix (Vektor als Spaltmatrix) zusammenfassen. Diese Objekte sind in einem eigenem, für sie optimalen, lokalem Koordinatensystem definiert. Dies bietet drei Vorteile: die Rotation um den lokalen Referenzpunkt ist einfacher zu berechnen, bei mehrmaligen Vorkommen vom selben Objekt kann eine Unterscheidung anhand des lokalen Ursprungs in der Szene erfolgen und komplexere Objekte können durch die zusammengesetzten Objekte mit eigenem Koordinatensystem manipuliert werden [23, S. 15 ff.] [21, S. 85].

3.4.1. Translation

Mithilfe eines Verschiebevektors T wird ein Punkt von P nach P_t bewegt. Die dazugehörige Berechnung in der homogenen Matrixnotation für Translation lautet [23, S. 15 ff.] [21, S. 86 ff.]:

$$P_t = T + P \qquad \Rightarrow \qquad \begin{bmatrix} P_{tx} \\ P_{ty} \\ P_{tz} \\ \mathbf{1} \end{bmatrix} = \begin{bmatrix} \mathbf{1} & 0 & 0 & t_x \\ 0 & \mathbf{1} & 0 & t_y \\ 0 & 0 & \mathbf{1} & t_z \\ 0 & 0 & 0 & \mathbf{1} \end{bmatrix} \cdot \begin{bmatrix} P_x \\ P_y \\ P_z \\ \mathbf{1} \end{bmatrix}$$

Abbildung A.9 zeigt ein Beispiel für eine Verschiebung um -12,4 Einheiten in x-Richtung, um -3,2 Einheiten in y-Richtung und um 0,5 Einheiten in z-Richtung. Das Ergebnis steht in der jeweiligen Matrixzeile.

3.4.2. Skalierung

Die Skalierung beschreibt die Streckung eines Objektes, wenn der Skalierungsfaktor $S > 1$ ist. Wenn der Skalierungsfaktor $S < 1$ ist, liegt eine Stauchung vor. Skalierung findet entlang der Koordinatenachsen statt. Bei proportionaler Skalierung gilt $s_x = s_y = s_z$. Faktoren im negativen Bereich sind ebenfalls möglich, welche eine Spiegelung an der jeweiligen Achse bewirkt. Der Wert 0 ist nicht zulässig, da dies eine Reduzierung der Dimension verursacht. Die dazugehörige Berechnung in der homogenen Matrixnotation für Skalierung lautet [23, S. 15 ff.] [21, S. 86 ff.]:

$$P_s = S \cdot P \qquad \Rightarrow \qquad \begin{bmatrix} P_{sx} \\ P_{sy} \\ P_{sz} \\ \mathbf{1} \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & \mathbf{1} \end{bmatrix} \cdot \begin{bmatrix} P_x \\ P_y \\ P_z \\ \mathbf{1} \end{bmatrix}$$

Die Skalierung im Beispiel auf der Abbildung A.9 beträgt 14,9. Zur besseren Handhabung kann mit der $Axis\text{-}Node$ das Skalierungszentrum verschoben werden.
Soll die Skalierung um einen Festpunkt F ausgeführt werden, welche als globale Deformation bezeichnet wird, ändert sich die Berechnung in der homogenen Matrixnotation wie folgt [23, S. 15 ff.] [21, S. 86 ff.]:

$$\begin{bmatrix} F_{sx} \\ F_{sy} \\ F_{sz} \\ \mathbf{1} \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ f_x \cdot (1 - s_x) & f_y \cdot (1 - s_y) & f_z \cdot (1 - s_z) & \mathbf{1} \end{bmatrix}$$

3.4.3. Scherung

Scherung bezeichnet die Änderung der Koordinaten einer Achse und gehört mit zur globalen Deformation. Sie wird bei der perspektivischen Verzerrung eingesetzt. Die dazugehörige Berechnung in der homogenen Matrixnotation für die Scherung in X-Richtung, Y-Richtung und Z-Richtung lautet [23, S. 15 ff.] [21, S. 86 ff.]:

$$\begin{bmatrix} \mathbf{1} & f_y & f_z & 0 \\ 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & \mathbf{1} \end{bmatrix} \quad \begin{bmatrix} \mathbf{1} & 0 & 0 & 0 \\ f_x & \mathbf{1} & f_z & 0 \\ 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & \mathbf{1} \end{bmatrix} \quad \begin{bmatrix} \mathbf{1} & 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 \\ f_x & f_y & \mathbf{1} & 0 \\ 0 & 0 & 0 & \mathbf{1} \end{bmatrix}$$

3.4.4. Spiegelung

Wie bei der Skalierung bereits erwähnt, lässt die Spiegelung eines Objektes an einer beliebigen Ebene die jeweiligen Ebenenkoordinaten unverändert und invertiert die entsprechende Raumkoordinate. Die dazugehörige Berechnung in der homogenen Matrixnotation für Spiegelung an der y-z-Ebene, x-z-Ebene und x-y-Ebene lautet [23, S. 15 ff.] [21, S. 86 ff.]:

$$\begin{bmatrix} \mathbf{-1} & 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & \mathbf{1} \end{bmatrix} \quad \begin{bmatrix} \mathbf{1} & 0 & 0 & 0 \\ 0 & \mathbf{-1} & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & \mathbf{1} \end{bmatrix} \quad \begin{bmatrix} \mathbf{1} & 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & \mathbf{-1} & 0 \\ 0 & 0 & 0 & \mathbf{1} \end{bmatrix}$$

3.4.5. Rotation

Um Objekte im Raum drehen zu können, muss man eine Rotationsachse festlegen. Fällt die gewählte Rotationsachse auf eine Koordinatenachse vereinfacht sich die Berechnung in der homogenen Matrixnotation für Drehung um x-, y- und z-Achse wie folgt [23, S. 15 ff.] [21, S. 86 ff.]:

$$P_r = R \cdot P$$

1	0	0	0
0	**cos**ϕ	**-sin**ϕ	0
0	**sin**ϕ	**cos**ϕ	0
0	0	0	**1**

cosϕ	0	**sin**ϕ	0
0	**1**	0	0
-sinϕ	0	**cos**ϕ	0
0	0	0	**1**

cosϕ	**-sin**ϕ	0	0
sinϕ	**cos**ϕ	0	0
0	0	**1**	0
0	0	0	**1**

Eine mathematisch positive Winkeländerung bewirkt eine Drehung gegen den Uhrzeigersinn. Wenn die Rotationsachse nicht mit einer Koordinatenachse übereinstimmt, existieren verschiedene Varianten der Berechnung, die im Laufe der Zeit entwickelt wurden. Eine Möglichkeit ist, die Drehung mit fünf hintereinander geschalteten Transformationen aufzulösen. Hierbei entsteht jedoch im ungünstigsten Fall das Phänomen Gimbal Lock, wobei zwei Koordinatenachsen zufällig zur Deckung kommen. Aus dieser Drehposition kann nicht mehr weg rotiert werden. Um das Phänomen zu umgehen, kann man sich der Mathematik der Quaternionen bedienen oder auf einen Ansatz von Walter-Dieter Klix zurückgreifen [21, S. 89 ff.].
Zur besseren Handhabung kann mit der $Axis\text{-}Node$ das Rotationszentrum verschoben werden.

KAPITEL 4

Visualisierung

Für die Anzahl der Polygone, die eine Szene beschreiben, gibt es keine obere Grenze. Um Ressourcen zu sparen, wurden zur Eingrenzung verschiedene Prozesse entwickelt. Zum Beispiel die Begrenzung des Sichtfeldes und das Entfernen von verdeckten Oberflächen stehen mit Hilfe des Z-Buffer zur Verfügung. Dieser Prozess wird im späteren Verlauf auch für die Schattierung und den Schattenwurf verwendet [23, S. 193]. Abbildung A.10 zeigt die verfügbaren Optionen bzw. Nodes, um die Visualisierung zu beeinflussen. Einige dieser Prozesse werden im folgenden Kapitel vorgestellt.

4.1. Clipping

3D-Clipping oder auf Deutsch Zuschneiden wird benutzt, um nicht alle Objekte einer Szene auf die Projektionsebene zu projizieren. Zuerst werden minimale und maximale Koordinaten definiert, die die endliche Projektionsebene aufspannen. Diese Koordinaten werden mit u für die x-Achse und v für die y-Achse bezeichnet. Es werden ebenfalls vor und hinter dem Beobachterstandpunkt Grenzen festgelegt. Diese Grenzen bilden einen Ausschnitt des Raumes. Alle Objekte außerhalb des Volumens werden nicht berechnet. Facetten, die über die Grenzen hinausgehen, werden beschnitten und bekommen Hilfskanten. Bei der Parallelprojektion hat das Volumen die Form eines Quaders. Wie in Abbildung 4.1 zu sehen ist, hat das Volumen bei der Zentralprojektion die Form eines vierseitigen Pyramidenstumpfes. Für die Zentralperspektive ist es günstiger, den sogenannten Öffnungswinkel des Pyramidenstumpfes anzugeben, statt die Projektionsebene zu definieren. Diese Technik wird Field of View bezeichnet [3, S. 37 ff.]. Ventuz verwendet standardmäßig die Field of View Technik mit einem Winkel von 45°, das entspricht einer Objektivbrennweite von 50mm. Mit der $Projection\text{-}Node$ aus Abbildung A.7 können diese Parameter verändert werden.

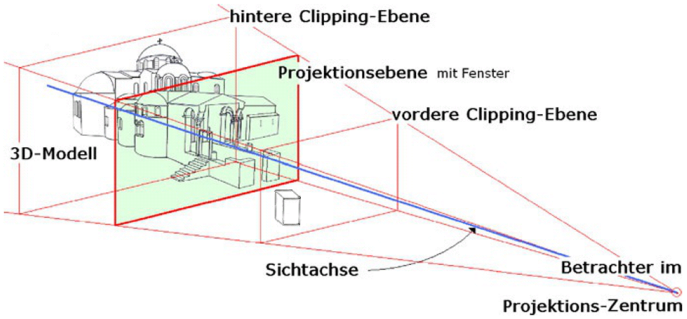


Abbildung 4.1.: Veranschaulichung des Zuschneidens bei der Zentralprojektion [21, S. 175]

4.2. Maleralgorithmus

Der Maleralgorithmus ist das einfachste Verfahren und wird direkt auf die angelegte Hierarchie angewendet. Objekte, die in der Liste oben stehen, werden zuerst abgebildet. Nachfolgende Objekte überdecken die bereits projizierten Flächen. Es ist in der Arbeitsweise einem Künstler der Malerei nachempfunden. Das Malen erfolgt ebenfalls in einer Reihenfolge. Es wird vom Hintergrund zum Vordergrund schichtweise gemalt. In Abbildung A.11 wird das Verfahren deutlich. Der Kubus wird zuerst projiziert, danach folgt das Zahnrad, welches den Kubus durchdringt. Der Maleralgorithmus hat einen entscheidenden Nachteil. Das Durchdringen einer Facette durch eine andere Facette wird nicht eindeutig projiziert [21, S. 188 ff.] [10, S. 158 ff.].

4.3. Z-Buffer Algorithmus

Mit Hilfe eines zusätzlichen Speichers, des Z-Buffers, können die Nachteile des Maleralgorithmus umgangen werden. In diesem Speicher wird für jedes Pixel des Ausgabegerätes die Tiefeninformation gespeichert. Dabei wird für jedes Objekt nacheinander geprüft, ob es näher an der vorderen Begrenzungsfläche des Sichtvolumens liegt, als das zuletzt geprüfte Objekt. Der Vergleich bezieht sich auf das Intervall zwischen vorderer und hinterer Begrenzungsfläche. Wenn das Objekt näher ist, wird der entsprechende Speicher mit einem neuen Farbwert und Z-Wert beschrieben.

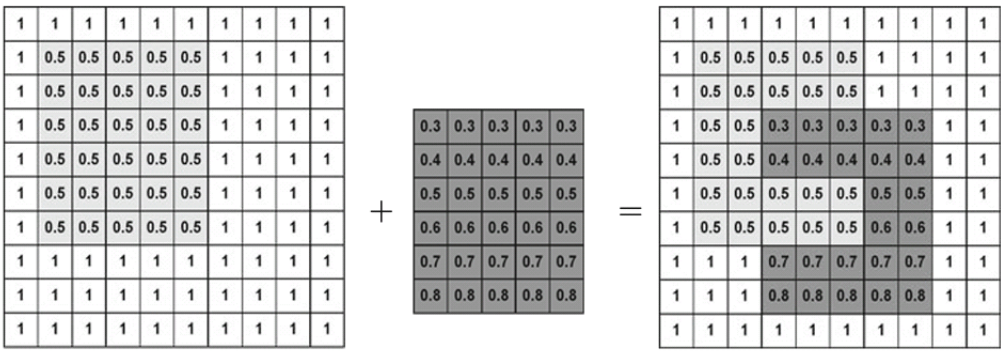


Abbildung 4.2.: Beispiel eines 10x10 Z-Buffer Speichers
[10, S. 161]

Für das Beispiel in Abbildung 4.2 würde der gesamte Speicher mit dem Wert 1 und Farbe Weiß für die hintere Begrenzung initialisiert. Für die vordere Begrenzungsfläche gilt der Wert 0. Oben links im Beispiel befindet sich auf der Hälfte des Sichtvolumens ein Objekt. Im nächsten Schritt wird ein weiteres Objekt projiziert. Dieses ist nach hinten geneigt und schneidet einen Teil des vorhergehenden Objektes. Zum besseren Verständnis wurden Werte zwischen 0 und 1 gewählt. Normalerweise verwendet man 16 Bit oder 32 Bit lange Zahlen. Wobei mit ganzen Zahlen gearbeitet wird, da die Verarbeitung schneller als mit Gleitkommazahlen ist.
In Abbildung A.12 sieht man die Auswahl der $Z\text{-}Testing\text{-}Node$ mit unterschiedlichen Vergleichsoperanden. Die Vergleichsoperation gibt einen booleschen Wert “wahr“ oder “falsch“ zurück.

Bei dem Rückgabewert “wahr“ wird der Speicher neu beschrieben und bei “falsch“ findet keine Änderung statt. “Never “ setzt alle Vergleiche auf “falsch“ .“Less “ ersetzt nur kleinere Werte als vorher. “Equal “ ersetzt nur gleiche Werte. “Greater “ ersetzt nur größere Werte als vorher. “NotEqual “ ersetzt nicht gleiche Werte. Bei “Always “ werden alle Vergleiche auf “wahr“ gesetzt.
Der Z-Buffer bietet gegenüber dem Maleralgorithmus folgende Vorteile:
Sich durchdringende Objekte werden mit der Genauigkeit einer Pixelfläche berechnet. Die Z-Werte können für die Schattierberechnung benutzt werden.
Nachteilig ist, dass transparente Oberflächen nicht korrekt berücksichtigt werden. Dies kann bei Animation mit bewegten transparenten Oberflächen negativ auffallen. Für diesen Fall bietet der Ventuz Designer die $Z\text{-}Sort\text{-}Node$, welche permanent den Z-Wert der lokalen Koordinaten vergleicht und Objekte aus der angegebenen Liste anordnet.
Die Leistungsfähigkeit ist abhängig von der Anzahl der zu füllenden Bildpunkte und dem verfügbaren Speicher. Durch die rasante Speicherentwicklung hat sich der Algorithmus in den letzten Jahren durchgesetzt [10, S. 159 ff.][21, S. 194 ff.].

4.4. Rückseitenentfernung

Das Objekt wird in Vorderseite und Rückseite von der Position des Betrachters unterteilt. Die Facetten, welche vom Objekt selbst verdeckt werden, werden entfernt. So können bis zu 50% des Rechenaufwandes eingespart werden. Dieser Algorithmus wird Back-Face-Culling, zu deutsch Rückseitenentfernung, genannt.
Zuerst wird zu jeder Facette die Normale berechnet, welche orthogonal auf ihr steht. Diese wird später ebenfalls für die Schattierverfahren benutzt. Zur Überprüfung, ob alle Normalen nach außen zeigen, wird aus der Summe aller Eckpunkte ein Hilfspunkt im Inneren gemittelt. Ein Vergleich zwischen dem Vektor “Eckpunkt $\rightarrow$ Hilfspunkt“ und der Flächennormalen zeigt an, ob die Normale richtig orientiert ist oder um 180° gedreht werden muss [21, S. 177 ff.].
Eine andere Möglichkeit die Normalen festzulegen ist, diese in der Datenstruktur mit Hilfe der Durchlaufrichtung der Eckpunkte zu orientieren. In Abbildung 2.3 würde die Facettennormale “$e_4\ e_3\ e_2\ e_1$“ nach außen zeigen, da die Nummerierung gegen den Uhrzeiger läuft. Eine Nummerierung mit “$e_1\ e_2\ e_3\ e_4$“ zeigt nach innen, da die Zählfolge in Uhrzeigerrichtung läuft. Zum Schluss findet der Vergleich zwischen Blickrichtung und Normale statt. Ist der Winkel kleiner als 90°, wird die Fläche entfernt [23, S. 45 f.].
Die $RenderOptions\text{-}Node$ aus Abbildung A.10 bietet die Manipulation dieses einfachen Verfahrens an, das bei der Reduzierung von zu rendernden Facetten hilft. Im Ventuz Designer ist die Rückseitenentfernung standardmäßig aktiviert. Auf Abbildung A.13 erkennt man auf der linken Seite die Deaktivierung des Verfahrens.

KAPITEL 5

Beleuchtung

Die Beleuchtung in der Computergrafik basiert auf den physikalischen Gesetzen des Lichtes, wie in Abbildung 5.1 zu sehen ist. Das Licht unterliegt dem Wellen-Teilchen Dualismus. Für die Ausbreitung wird das Licht als eine Welle interpretiert. Findet eine Wechselwirkung mit Materie statt, betrachtet man das Licht als Teilchen. Die realen Eigenschaften von Licht sind kompliziert und lassen sich nicht in ihrer Gesamtheit mathematisch formulieren. Aus diesem Grund ist eine Vereinfachung der physikalischen Gesetze zur Reduktion des Rechenaufwandes notwendig [10, S. 215].
Um das Licht näherungsweise zu beschreiben, bedient man sich der Strahlenoptik, in der der geometrische Weg des Lichtes als Linie beschrieben wird. In der Theorie lassen sich viele optische Phänomene mit folgenden Grundprinzipien beschreiben [3, S. 271].

- Licht breitet sich in homogenen Medien geradlinig aus.
- Sich kreuzende Strahlen haben keinen Einfluss aufeinander.
- Der Strahlengang ist reversibel.
- Beim Übergang zwischen zwei Medien unterliegt das Licht dem Reflexions- und Brechungsgesetz.
- Trifft Licht auf eine Oberfläche wird es in Anteilen absorbiert, remittiert und transmittiert.

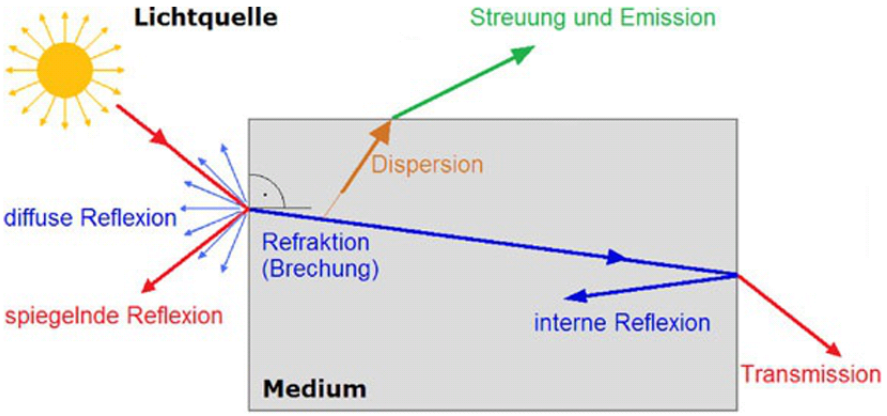


Abbildung 5.1.: Verlauf eines Lichtstrahls
[21, S. 212]

In der Computergrafik sind zwei Kategorien entstanden, die unterschiedliche Anforderungen an die Bildqualität stellen. Die lokalen Beleuchtungsmodelle, welche mit möglichst einfacher Realisierung zufriedenstellende Ergebnisse generieren und die globalen Beleuchtungsmodelle, bei denen die physikalischen Vorgänge exakt modelliert werden, um das realistischste Ergebnis zu erreichen. Der Aufwand für realitätsnahen Darstellung bedingt, dass die globalen Beleuchtungsmodelle für Echtzeit bzw. Interaktivität ungeeignet sind [3, S. 270].

5.1. Lichtquellen

Zum Modellieren einer Szene werden Lichtquellen benutzt, die die Szene wie gewünscht beleuchten. In diesen Abschnitt werden die gängigen Lichtquellen vorgestellt [3, S. 277] [21, S. 213 ff.]:

- **ungerichtete Lichtquelle:** Sie sorgt für eine gewisse Grundhelligkeit auf nicht direkt beleuchteten Facetten. Die Bezeichnung ambientes Licht oder Hintergrundlicht wird ebenfalls verwendet. Sie entsteht durch Reflexion an matten Oberflächen.
- **gerichtete Lichtquelle:** Die Lichtquelle ist zum Objekt unendlich entfernt und die Lichtstrahlen verlaufen parallel zueinander. Dieser Sachverhalt führt zu einer vereinfachten Berechnung. Der Vektor von der Facette zur Lichtquelle muss nur einmal berechnet werden und wird für die restlichen Facetten übernommen.
- **punktförmige Lichtquelle:** Dies ist eine Kugellichtquelle mit sehr kleinem Durchmesser. Sie strahlt gleichmäßig Licht in alle Richtungen aus. Die Intensität des Lichtes kann optional mit einer entfernungsabhängige Abschwächung versehen werden. Grundlage ist die Verteilung des Lichtes auf einer Kugelschale. Daraus folgt, dass die Intensität umgekehrt proportional zum Abstand im Quadrat abnimmt.
- **Scheinwerfer:** Sie sind punktförmige Lichtquellen mit einer Begrenzung des Raumwinkels. Der Öffnungswinkel des entstehenden Lichtkegels kann beeinflusst werden. Sie können zusätzlich mit Entfernungs- und Randabschwächung versehen werden.
- **Fläche:** Sie ist eine Sonderform. Es werden mehrere punktförmige Lichtquellen zueinander benachbart. Dies führt zu einer weichen Ausleuchtung von Objekten, jedoch verursacht sie erheblichen Rechenaufwand.

Im Ventuz Designer befindet sich standardmäßig eine gerichtete Lichtquelle im Beobachterstandpunkt, welche die Blickrichtung der Strahlrichtung entspricht. Wird eine Lichtquelle in die Szene gesetzt, entfällt die Lichtquelle aus dem Beobachterstandpunkt. Es stehen drei verschiedene Lichtquellen zur Verfügung, die in Abbildung A.15 zu sehen sind. Mit der $LightingGroupNode$ kann eingeschränkt werden, welche Lichtquellen auf welche Objekte wirkt.

5.2. Lokale Beleuchtungsmodelle

Die lokalen Beleuchtungsmodelle wurden in den 70er Jahren entwickelt und haben sich bis heute durchgesetzt. Sie bestehen aus zwei Stufen. In der ersten Stufe wird anhand des Reflexionsmodells die Lichtintensität berechnet, die an einer Normalen auf einer Facette herrscht. Diese Berechnung findet im Weltkoordinatensystem statt. Nachfolgend wird das phongsche Reflexionsmodell erläutert, das als De-Facto-Standard für die Berechnung der Intensität gilt.
In der zweiten Stufe wird die Färbung der Pixel bestimmt, die die entsprechende Facette als kleine Teilflächen abbilden. Die Pixel werden mit Schattieralgorithmen eingefärbt, welche in Abschnitt 5.2.1. erläutert werden. Diese Färbung findet im normalisierten Koordinatensystem statt. Die Intensität wird als Summe von vier Beleuchtungskomponenten gebildet, die im Folgenden näher erläutert werden [10, S. 224 ff.] [23, S. 198 ff.] [21, S. 215 ff.]:

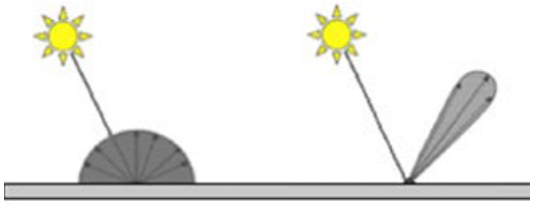


Abbildung 5.2.: Veranschaulichung der diffusen und spekularen Beleuchtungskomponente [21, S. 222]

- **emessive:** Damit werden die selbstleuchtenden Lichtanteile eines Objektes bezeichnet. Der emessive Anteil hat keinen weiteren Einfluss auf andere Objekte. Mit dem emessiven Lichtanteil kann man ein Objekt mit einem gewünschten Farbwert belegen.

- **ambiente:** Damit werden die Anteile des ungerichteten Streulichtes bezeichnet und simulieren eine indirekte Beleuchtung. Sie kann Facetten einfärben, die nicht direkt von Lichtquellen beschienen werden, aber im Blickfeld des Betrachters liegen. Ansonsten würden die Facetten schwarz gerendert werden und wären somit nicht sichtbar. Mit ihr kann eine Grundhelligkeit der Szene bestimmt werden und berechnet sich mit folgender Formel:

  $$I = I_a \cdot k_a$$

  Die Intensität des ambienten Lichtes wird mit dem Reflexionskoeffizienten $k \leq 1$ multipliziert, der die Oberflächeneigenschaften für ambientes Licht beschreibt.

- **diffuse:** Damit werden die Anteile des gerichteten Lichtes bezeichnet, welche ideal diffus reflektiert werden. Die Reflexion verteilt sich gleichmäßig auf die Form einer Halbkugel, wie in Abbildung 5.2 auf der linken Seite zu sehen ist. Damit ist die Reflexion unabhängig von der Blickrichtung des Betrachters. Die Intensität ändert sich je nach Einfallwinkel und wird mit folgender Formel berechnet:

  $$I = I_d \cdot k_d \cdot cos(\theta)$$

  Die Intensität des diffusen Lichtes wird mit dem Reflexionskoeffizienten $k \leq 1$ multipliziert und in Abhängigkeit seines Einfallwinkels θ gesetzt. k_d beschreibt die Oberflächeneigenschaften für die diffusen Lichtanteile.

- **spekulare:** Damit werden die Anteile des gerichteten Lichtes bezeichnet, die ideal spiegelnd reflektiert werden. Sie bewirkt eine spiegelnde Stelle auf dem Objekt in der man die Farbe der Lichtquelle sieht. Da die ideale Reflexion in der Natur nicht existiert, wird sie zur unvollkommenen spiegelnden Reflexion erweitert. Das Licht wird in einen Reflexionskegel aufgespalten, wie in Abbildung 5.2 auf der rechten Seite zu sehen ist. Dies bewirkt eine blickwinkelabhängige Reflexion und berechnet sich mit folgender Formel:

  $$I = I_s \cdot k_s \cdot cos^a(\varphi)$$

Die Intensität des spekularen Lichtes wird mit dem Reflexionskoeffizienten $k \leq 1$ und mit einer $cos^a(\varphi)$ Abklingfunktion multipliziert. θ bezeichnet den Winkel, der zwischen dem Betrachter und Reflexionsrichtung des Lichtes herrscht. a ist der Wert für die spiegelnden Oberflächeneigenschaften. k_d beschreibt die Oberflächeneigenschaften für die spekularen Lichtanteile.

Im Laufe der Zeit wurden weitere Berechnungsmethoden entwickelt, die die phongsche Berechnung erweitern oder verbessern.
Mit der $Material\text{-}Node$ aus der Abbildung A.14 kann man die Reflexionskoeffizienten beeinflussen. Die Option $Sharpness$, wie in Abbildung A.16 zu sehen ist, nimmt dabei Einfluss auf die spiegelnde Oberflächeneigenschaft. Ein Wert $a < 25$ simuliert eine raue und ein Wert $a > 25$ glatte Oberflächen. Die jeweiligen Intensitätswerte werden der Lichtquelle entnommen.

5.2.1. Schattierung

Nachdem die Intensitätswerte errechnet wurden, färbt der Schattieralgorithmus die entsprechenden Pixel ein, die auf die Facetten projiziert werden. Dieses Verfahren wird im englischen als Shading bezeichnet und vervollständigt die lokale Nachahmung der Beleuchtungsverhältnisse. Es wurden drei unterschiedliche Schattierungsverfahren entwickelt, die im Nachfolgenden vorgestellt werden. Sie unterscheiden sich jeweils in ihrer Qualität und dem verwendetem Rechenaufwand in steigender Reihenfolge [21, S. 234 ff.] [23, S. 289 ff.].

- **Flat:** Flat-Shading ist die einfachste und schnellste Methode. Sie erzeugt eine konstante Einfärbung der Pixel, die auf die entsprechende Facette fallen. Zuerst werden die Normalen auf den Facetten bestimmt, danach findet die Berechnung mit Beleuchtungsmodell statt und zum Ende werden die Pixel nach der Intensität eingefärbt. Da die Intensität nur einmal pro Facette berechnet wird, werden viele Pixel mit dem gleichen Farbwert berechnet und die Kanten bzw. Übergänge zwischen den Facetten sind erkennbar.

- **Gouraud:** Gouraud-Shading benutzt die Ecknormalen zur Berechnung der Intensität. Es wird der Umstand ausgenutzt, dass eine Ecke an mehrere Facetten grenzt. Die Farbwerte für die Pixel, die auf eine Facette fallen, werden von Ecke zu Ecke interpoliert. Damit erreicht man weichere Übergänge. Probleme entstehen, wenn Lichtquellen nah am Objekt liegen. Außerdem sind Glanzlichter, durch ihre vielen Details, schwer darstellbar. Die gesamte Darstellung kann durch Tesselation,wie in Kapitel 2.3 vorgestellt, verbessert werden. Dies führt jedoch zu erhöhten Rechenaufwand.

- **Phong:** Phong-Shading benutzt ebenfalls die Ecknormalen für die Berechnung der Intensität. Es wird jedoch für jeden Pixel, der auf eine Facette fällt, zusätzlich eine weitere Normale zwischen den Ecknormalen interpoliert. Danach wird für jede Normale die Beleuchtungsberechnung angewendet und die Pixel eingefärbt. Dieses Verfahren bietet eine exakte Intensitätsverteilung und eine korrekte Darstellung von Glanzlichtern auf den Facetten. Die Bildqualität ist im Verhältnis zu Gouraud- oder Flat-Shading besser. Nachteilig ist, dass es aufwändigere Berechnungen und damit mehr Rechenzeit benötigt.

Im Ventuz Designer werden Flat und Gouraud verwendet. In Abbildung A.17 ist die erste Kugel mit der $Color\text{-}Node$ kombiniert, welche die Kugel mit einem Farbwert festlegt. Für einen

dreidimensionalen Eindruck ist die $Node$ ungeeignet, da die Kugel als Kreis erscheint. An der zweiten Kugel ist das Gouraud Verfahren zu sehen, welcher in Ventuz als vordefinierter Standard eingesetzt wird. Für die dritte und vierte Kugel wurde die $Material\text{-}Node$ verwendet. Bei ihr kann von Gouraud auf Flat Verfahren umgeschaltet werden. Mit dieser $Node$ lassen sich zusätzlich die Farbwerte für emessiv, ambient, diffus und spekular beeinflussen.

5.3. Globale Beleuchtungsmodelle

Die globalen Beleuchtungsmodelle berücksichtigen das emittierte Licht und das transmittierte Licht von Lichtquellen und Oberflächen. Sie stellen die realitätsnahe Wiedergabe des Lichtes in einer Szene in den Vordergrund. Dabei werden die physikalischen Vorgänge so exakt wie möglich nachempfunden. Es gibt zwei unterschiedliche Ansätze für globale Beleuchtungsverfahren, das Strahlungsverfahren Radiosity und das Strahlverfolgungsverfahren Raytracing. In Ventuz werden globale Beleuchtungsverfahren nicht unterstützt, können aber durch Skripte implementiert werden. Aus diesem Grund werden die unterschiedlichen Verfahren kurz erläutert [6, S. 141] [3, S. 278 f.] [21, S. 237 ff.]:

- **Radiosity:** Das Radiosity Verfahren basiert auf dem Energieerhaltungssatz und der Annahme das alle Oberflächen ideal diffuse Reflektoren sind. Das auftreffende Licht, welches von einer Facette nicht absorbiert wird, wird von ihr an benachbarte Facetten abgegeben. Für diese Berechnung benötigt man mehrere Renderdurchläufe, welche als Iterationsschritte bezeichnet werden. Im ersten Schritt wird berechnet, wie viel Licht auf eine Facette trifft und im zweiten Schritt, wie viel Licht von ihr abgegeben wird. In Abbildung 5.3 sind Bilder nach einer gewissen Anzahl von Iterationsschritten zu sehen. Im letzten Bild ist erkennbar, dass sich die Decke und die grüne Kugel gegenseitig beeinflussen und die rote Farbe der linken Wand die gesamte Szene dominiert. Ebenso werden weitere Materialeigenschaften für alle Facetten in einer Szene benötigt. Ein Vorteil dieses Verfahrens ist, dass die Berechnung unabhängig vom Beobachterstandpunkt ist, weil die Energieverteilung für alle Facetten berechnet wird. Allein die Verdeckungsalgorithmen müssen zusätzlich angewendet werden.

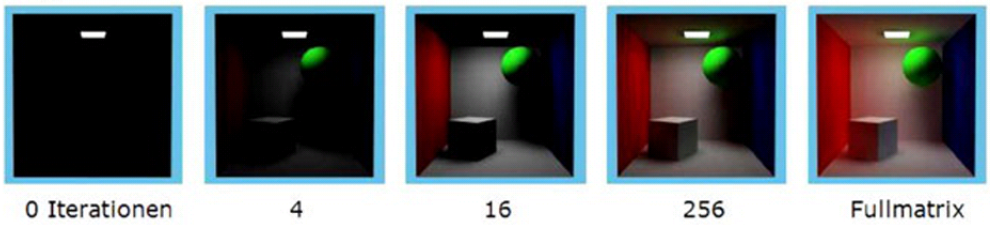


Abbildung 5.3.: Veranschaulichung des Radiosity Verfahrens mit Iterationsschritten [21, S. 256]

- **Raytracing:** Das Raytracing Verfahren basiert auf den optischen Gesetzen der idealen Spiegelung und Brechung. Das Prinzip ist die Verfolgung der Lichtstrahlen einer Lichtquelle bis zum Auftreffen im Beobachterstandpunkt. Mathematisch gesehen ist die Wahrscheinlichkeit diesen Punkt zu treffen nicht möglich. Deshalb wird das Prinzip rekursiv

berechnet. Ausgehend vom Beobachterstandpunkt wird durch ein Raster in die Szene eine Anzahl von Primärstrahlen gesendet. Das Raster entspricht der Pixelauflösung des Wiedergabesystems und der Primärstrahl schneidet je ein Pixel. Trifft der Primärstrahl auf eine Oberfläche, so spaltet sich dieser in zwei neue Strahlen, ein Reflexions- und Brechungsstrahl. Diese entstandenen Strahlen werden weiterverfolgt bis bestimmte Bedingungen erfüllt sind, dann wird der Prozess beendet. Eine Bedingung ist die Rekursionstiefe, welche die Anzahl der Spaltungen angibt. In Abbildung 5.4 ist das Raytracingverfahren zu sehen. n stellt die Normale der Facette dar. r ist der Reflexionsstrahl, t der Brechungsstrahl und I der Strahl zur Lichtquelle.

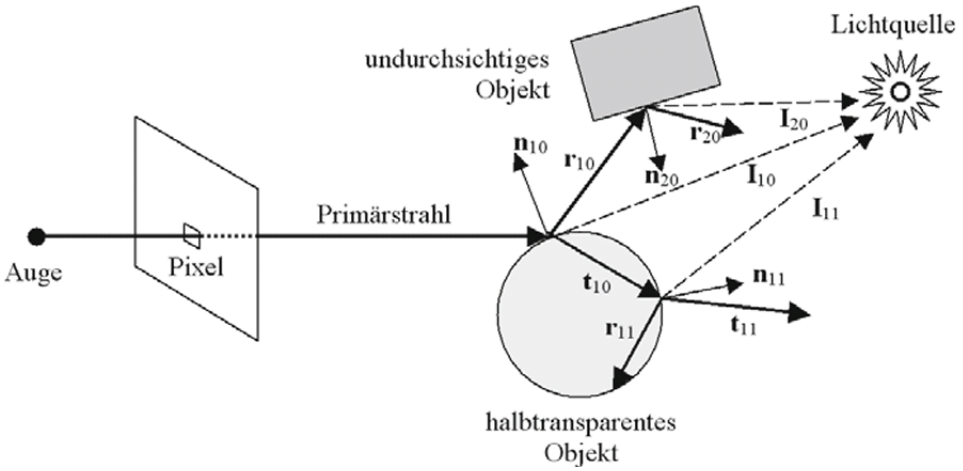


Abbildung 5.4.: Veranschaulichung des Raytracing Verfahrens
[10, S. 225]

5.4. Schattenwurf

Die zweidimensionale Abbildung einer Raumsituation ist nicht eindeutig. Erst der Schattenwurf stellt zusätzliche Informationen zur Verfügung. Durch Kenntnis der Schatten sind die räumlichen Beziehungen zueinander rekonstruierbar [8, S. 41]. Die lokalen Beleuchtungsmodelle berücksichtigen keinen Schattenwurf. Die Untersuchung von Lichtstrahlen und ob diese von Objekten unterbrochen werden, ist kein Bestandteil ihrer Berechnung. In der Realität unterscheidet man zwischen zwei Schattenarten, die harten Schatten erzeugt durch punktförmige Lichtquellen und die weichen erzeugt durch flächige Lichtquellen. Die Komplexität kann durch transparente Objekte weiter zunehmen und anhand dieser Komplexität werden die Schattenwurfalgorithmen eingeteilt. Ventuz unterstützt die Generierung des Schattenwurfes nicht. Zur Vollständigkeit soll kurz ein einfaches Verfahren vorgestellt werden, mit dem Schatten simuliert werden können.
Mit der Textur und den Mapping Verfahren, die in Kapitel 6 vorgestellt werden, kann man einfache harte Schatten erzeugen. Dieses Verfahren wird im englischen als Shadow-Mapping bezeichnet. Im ersten Schritt wird die Szene aus Sicht der Lichtquelle gerendert und erzeugte Z-Werte in der Textur abgelegt. Im zweiten Schritt wird die Szene aus Sicht des Beobachterstandpunktes gerendert. In diesen Renderprozess fließt die Schattentextur mit ein. Die aus beiden Renderdurchläufen gewonnenen Z-Werte werden miteinander verglichen, wie in Abbil-

dung 5.5 zu sehen ist. Ist der Z-Wert vom Beobachterstandpunkt Z_P größer als der Z-Wert der

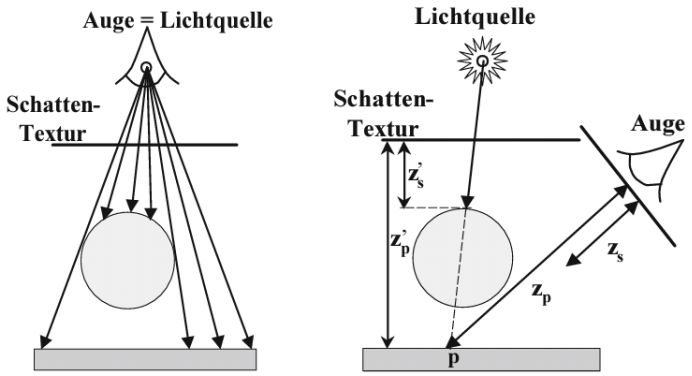


Abbildung 5.5.: Veranschaulichung des Verfahrens für eine Schattentexture
[10, S. 368]

Schattentextur Z_S, werden für die Beleuchtungsrechnung nur die emessiven und ambienten Komponenten einbezogen. Diese Methode ist echtzeitfähig und kann auf aktuellen Grafikkarten eingesetzt werden [10, S. 362 ff.].

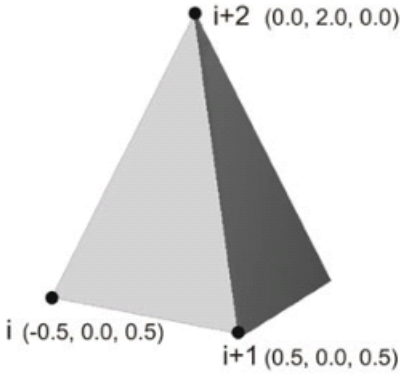
KAPITEL 6

Textur

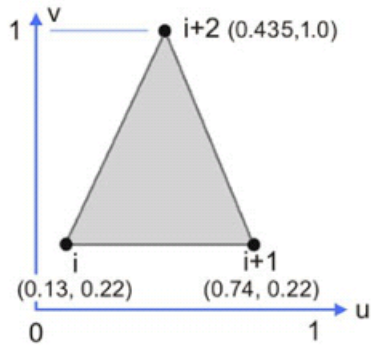
In der Natur besitzen reale Oberflächen von Körpern eine regelmäßige oder unregelmäßige Struktur bzw. Muster. Mit den bisher vorgestellten Beleuchtungsmethoden und der $Color\text{-}Node$ werden die Farben direkt aufgebracht. Um die Oberflächenstruktur realistisch darzustellen, wurden Verfahren entwickelt, die die Oberflächenstrukturen simulieren und das Objekt mit einem Muster überziehen. Der Ventuz Designer stellt dafür verschiedene $Nodes$ zur Verfügung, die in Abbildung A.18 zu sehen sind. Das Überziehen wird im englischen Sprachgebrauch Mapping und das Muster Texture genannt. Der Begriff Textur stammt vom lateinischen textura, was Gewebe bedeutet. Die Textur kann in zwei Kategorien, die diskrete und die prozedurale, eingeteilt werden [10, S.266 f.].
In Abbildung 6.1 werden die Vorgänge veranschaulicht, die die Texture beim Aufbringen durchläuft. Erwähnenswert ist, dass die Textur in ein eigenes Koordinatensystem in Form eines Einheitsquadrates überführt wird. Dies vereinfacht die Zuordnung von Texel auf die Facetten. Analog zum Pixel gibt es das Kunstwort Texel von Texture Element. Die Koordinatenachsen tragen die Bezeichnung u für x-Achse, v für y-Achse und haben das Intervall [0,1]. Die Definition über dieses Intervall hinaus, lässt Techniken des Kachelns und Spiegelns zu. Da die Texel nicht eins zu eins auf die Facetten übertragen werden können, werden manche Farbwerte durch Interpolation gewonnen [21, S.293 ff.].
Zu Beachten ist, das beim Import von nicht Ventuz eigenen Objekten die U/V-Koordinaten der Facetten im Modellierprogramm festgelegt werden müssen.

6.1. Diskrete Texturen

Diskrete Texturen sind zweidimensionale Rastergrafiken, die in unterschiedlichen Formaten wie zum Beispiel *.jpeg, *.png, *.TIFF gespeichert sind. Bei den Formaten unterscheidet man in unkomprimiert, verlustfreie und komprimiert, verlustbehaftete. Im Grafikspeicher werden die Rastergrafiken unkomprimiert verarbeitet, was viel Speicherplatz benötigt. Hierbei ist jedem Bildpunkt ein konkreter Farbwert zugewiesen.
Die einzelnen Speicherzellen sind in großen Gruppen zusammengeschaltet und die Adressierung findet nach dem Schema der Zweierpotenz statt. Aus diesem Grund empfiehlt sich die Auflösung der Grafik an dieses Schema anzulehnen. Eine Textur besteht zum Beispiel aus einer Auflösung von 1024 x 768 Pixel und einer Farbtiefe von 24 Bit. Die 768 Pixel sind aus einer Addition von 512 und 256 zusammengesetzt. 24 Bit Farbtiefe bedeutet 8 Bit bzw. 256 Werte für die Farbe Rot, 8 Bit für die Farbe Grün, 8 Bit für die Farbe Blau. Diese Farbauswahl wird RGB-Modell genannt und kann auf 32 Bit Grafiken erweitert werden, um zusätzlich 8 Bit für die Transparenz zur Verfügung zu stellen. Die 24 Bit Farbtiefe entspricht einem verfügbaren Spektrum von ca. 16,7 Millionen Farbnuancen.
Mit der Farbtiefe von 24 Bit und Anzahl von 1024 x 768 Pixel braucht die Textur eine Speicherkapazität von 18 MiBit oder die gebräuchliche Angabe von 2,25 MiB. Stellt man diese Textur



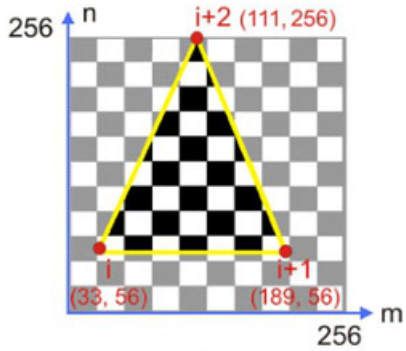
Objektraum
Das 3D-Modell wird mit einer der Projektionen projiziert. Für jeden Knoten i einer Facette sind damit seine Koordinaten (x_i, y_i, z_i) im Objektraum bekannt und auch seine Gerätekoordinaten (h_i, v_i) im Projektionssystem. Zusätzlich sind für jeden Knoten uv-Texturkoordinaten erforderlich, mit denen die Verbindung vom Knoten zum Texel hergestellt wird.



Parameterraum
Die Koordinaten der drei Facettenknoten werden in 2D-Koordinaten des Parameterraums überführt

$$(x_i, y_i, z_i) \Rightarrow (u_i, v_i)$$

und im Bereich [0,1] skaliert. Die Parameterkoordinaten (u_i, v_i) geben jetzt die relative Position des Knotens i in der Textur an.



Texturraum
Hier erfolgt der Übergang vom Parameter- in den Texturraum. Den relativen Parameterkoordinaten (u_i, v_i) werden ganzzahlige Texturkoordinaten (m_i, n_i) der Texel zugewiesen.

$$(u_i, v_i) \Rightarrow (m_i, n_i)$$

Für eine Texturgröße von 256×256 ergeben sich die Texturkoordinaten im Beispiel zu $(m_i, n_i) = 256 \cdot (u_i, v_i)$.

Bildraum
Infolge der räumlichen Darstellung ist die von den Pixelkoordinaten (h_i, v_i) gebildete Facette nicht affin mit derjenigen aus den (m_i, n_i)-Koordinaten des Texturausschnitts. Deshalb werden die Texturkoordinaten zunächst entlang der Facettenkanten, dann in ihrem Inneren linear interpoliert. Hierbei sind schräge auf gerade Linien (oder umgekehrt) abzubilden.

Abbildung 6.1.: Beispiel für eine Texture Pipeline
[21, S. 296]

auf einem Ausgabegerät dar, muss diese entsprechend der Bildwiederholrate periodisch verarbeitet werden. Bei einer Bildwiederholrate von 60 Hz, muss die Grafikkarte ca. 1,05 GiBit/s verarbeiten. Die Grafikkarte Quadro M4000 aus Kapitel 1 hat, laut Hersteller, eine Speicherbandbreite von 192 GBit/s und einen verfügbaren Speicher von 8 GB [17], welche für die Verarbeitung von mehreren Texturen ausgelegt ist.

6.2. Prozedurale Texturen

Prozedurale Texturen werden im Gegensatz zu diskreten Texturen aus mathematischen Funktionen generiert. Es können einfache Muster, wie die Maserung von Holz oder sich periodisch wiederholende Strukturen erzeugt werden. Bei komplexeren Mustern wie Marmorierung ist die mathematische Beschreibung schwierig. Die Beschreibung solcher Strukturen nimmt in der Mathematik das Gebiet der fraktalen Geometrie ein [21, S. 292].
Im Ventuz Designer kann auf die $TextureGenerator\text{-}Node$, welche auf Abbildung A.19 zu sehen ist, zurückgegriffen werden. Die $Node$ stellt fünf Algorithmen zur Erzeugung von Texturen zur Verfügung. Die Berechnung erfolgt vorab und die generierte Rastergrafik wird als diskrete Textur temporäre abgespeichert. Durch die Option “Seamless“ können Texturen ohne sichtbare Grenzen berechnet werden.

6.3. Mapping Verfahren

Mapping bezeichnet die Abbildung der Texture auf ein Objekt und ermöglicht die Simulation von verschiedenen Effekten wie globale Beleuchtung, Schattenwurf, Spiegelung und Struktur, ohne dabei aufwändige Berechnungen anzuwenden. In Abbildung A.20 werden verschiedene $Nodes$ vorgestellt, welche verschiedene Einstellmöglichkeiten für die Textur bereitstellen. Nachfolgend werden einige Punkte erläutert.
Im Ventuz Designer deklariert die Auswahl bei “Mode“ die Ausdehnung einer Textur. “T1D“ bedeutet, das die Textur eine Zeile bzw. ein langer Streifen ist. Mit “T2D“ wird die Verwendung einer zweidimensionalen Fläche angekündigt. Mit “T3D“ wird die Texture für einen Kubus ausgewählt und mit “T4D“ eine Textur, die sich über ein Volumen erstreckt.
Für die Festlegung der u,v,w-Achse im Parameterraum gibt es unterschiedliche Auswahlmöglichkeiten. Es gibt “Wrap“ , welche dem Aneinanderreihen von Texturen entspricht. Die Option “Mirror“ , welche die Textur in die entsprechende Richtung spiegelt. Die Option “Border“ , welche alle übrigen Facetten außerhalb der Texturkoordinaten mit definierter Farbe gefärbt werden und die Option “Clamp“ , welche die Facetten in das Interval [0,1] presst.
Unter “Type“ kann man die Technik auswählen, mit der die Texturkoordinaten generiert werden. “PassThru“ wird die gängige Methode bezeichnet, die in Abbildung 6.1 veranschaulicht ist. Weitere Techniken werden unter dem Begriff S-/O-Mapping erklärt.

6.3.1. S-/O-Mapping

Um die Texturkoordinaten zuzuordnen, unterteilt das S-/O-Mapping Verfahren die Berechnung in zwei weitere Schritte. Im ersten Schritt wird die Textur grob angepasst und auf ein einfaches dreidimensionales Zwischenobjekt projiziert. Dies wird als S-Mapping bezeichnet, wobei

S für Surface, aus dem Englischen für Oberfläche, steht. Im zweiten Schritt wird die Textur von diesem Zwischenobjekt auf das gewünschte Objekt übertragen. Dies wird als O-Mapping bezeichnet, wobei O für Object steht. Mit dieser Technik kann man die Parametrisierung von komplexen Oberflächen des Polygonennetzes umgehen [23, S.262 ff.].
In der $2DMapping\text{-}Node$ kann man neben der normalen Parametrisierung die Kugel oder den Kubus als Zwischenobjekt auswählen. Weitere Auswahlmöglichkeiten sind die “CameraSpace-Normal“, dabei werden die Flächennormalen in Augenpunktkoordinaten umgewandelt und als Texturkoordinaten benutzt, die “CameraSpacePosition“ , dabei werden die Eckpunkte in Augenpunktkoordinaten umgewandelt und als Texturkoordinaten benutzt und die “CameraSpaceReflectionVector“ , dabei wird ein Reflektionsvektor berechnet, danach in Augenpunktkoordinaten umgewandelt und als Texturkoordinaten benutzt.

6.3.2. Mip Mapping

Mip Mapping ist ein Verfahren, um die Qualität von Texturen vor der Abbildung zu erhöhen. Sie ähnelt dem Level-of-Detail Verfahren, wie in Kapitel 2.3 vorgestellt. Zur Originaltextur werden Kopien mit halbierender Größe erzeugt. Die um die Auflösung verringerte Textur wird verwendet, wenn der Pixelabstand viel kleiner als der Texelabstand ist [21, S. 303]. Die Kopien bilden eine sukzessive Folge bis genau oder annähernd 1x1 Texel erreicht sind. In Abbildung 6.2 ist eine MipMap veranschaulicht. Die n-te Verkleinerung wird als Mipmap-Level n bezeichnet. Mip ist eine Abkürzung für “multum in parvo“ , es kommt aus dem lateinischen und bedeutet “viele in einem“ . Sollte die ursprüngliche Textur nicht quadratisch sein, bleibt am Ende ein Streifen bzw. eindimensionale Textur übrig. Die MipMaps werden vorab berechnet und in temporäre diskrete Textur abgespeichert, um Rechenzeit zu sparen.
Vorteil dieses Verfahrens ist, dass durch geringen Aufwand Aliasing-Effekte vermieden werden. Dies ist ein Begriff aus der Signaltheorie für auftretende Fehler. Hierbei ist die Signalfrequenz höher als die Abtastfrequenz und führt zu einem Informationsverlust. Nachteilig ist der erhöhte Speicherbedarf [10, S. 283 ff.].

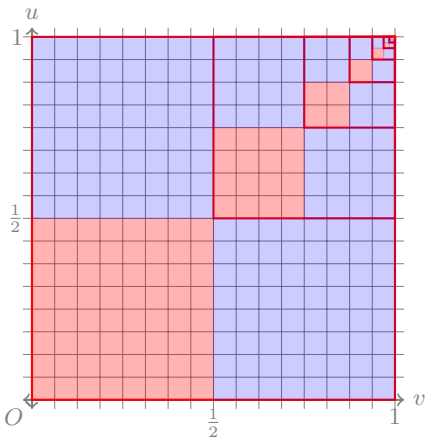


Abbildung 6.2.: Veranschaulichung eines MipMaps

6.3.3. Bump Mapping/Displacement Mapping

Bump stammt aus dem Englischen und heißt Delle oder Unebenheit. Mit ihr entsteht der Eindruck einer Oberflächenstruktur bzw. eines Reliefs, welches nicht im geometrischen Modell vorhanden ist.
Die Abbildung A.21 zeigt ein Beispiel aus Ventuz, auf welchem Bump Mapping deutlich zuerkennen ist. Durch Verdunklung und Aufhellung werden Vertiefung und Erhöhung simuliert. Um die Beleuchtungsverhältnisse zu verändern, werden die Normalen der Facetten manipuliert[3, S. 305 f.]. Die Simulation von Struktur funktioniert zufriedenstellend in Verbindung mit effektiven Schattierungsverfahren wie Phong oder Raytracing aus Kapitel 5.
Um die Normalen zu manipulieren, wird ein zweidimensionales Höhenfeld verwendet. Aus diesem werden die entsprechenden Höhenwerte h(u) für die Facetten entnommen. Die Höhenwerte werden zu den gegebenen Originalnormalen N(u) addiert. Somit entsteht aus der Originalkurve bzw. Fläche P(u) die Offsetkurve P'(u). Im nächsten Schritt werden mit Hilfe der Tangentialebene die Winkel der neuen "gestörten" Normalen N'(u) berechnet [21, S. 298 ff.]. Die Berechnung ist in Abbildung 6.3 zu sehen.
Ein entscheidender Nachtteil existiert bei diesen Verfahren. Schaut man auf das Objekt mit

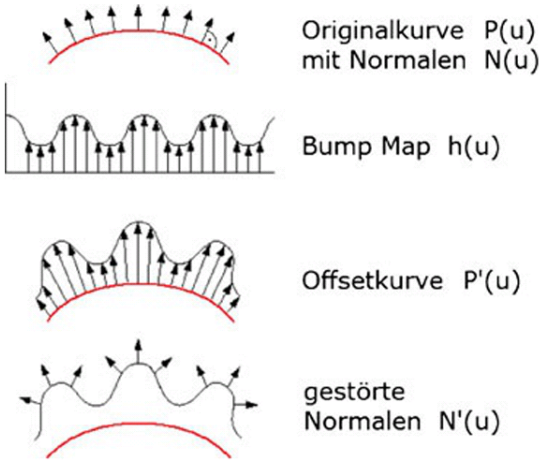


Abbildung 6.3.: Veranschaulichung des Bump Mapping Verfahrens
[23, S. 270]

einem sehr kleinen Winkel wirkt die Oberfläche dennoch glatt. Um diesen Nachteil zu beseitigen wurde das Displacement Mapping entwickelt. Displacement kommt aus dem Englischen und bedeutet Verschiebung oder Verrückung. Es ist ähnlich wie das Bump Mapping aufgebaut, jedoch wird entsprechend dem zweidimensionalen Höhenfeld die Objektgeometrie verändert. Durch Verschieben der Eckpunkte entlang ihrer Normalen verändert man das Polygonennetz. Wenn nicht genug Maschen zur Verfügung stehen, um das Relief zu erzeugen, kann zusätzlich durch Tesselation das Netz verfeinert werden. So entsteht auch bei flachen Betrachtungswinkeln eine unebene Struktur auf dem sonst glatten Objekt [21, S. 303].

6.3.4. Environment Mapping

Um die Umwelt bzw. Umgebung zu simulieren, wurde das Environment Mapping entwickelt. Auf der Innenseite eines virtuellen Körpers wird die Umgebung abgebildet, wobei sich der Be-

obachterstandpunkt mit im Inneren befindet. Vorrangig werden für diese Körper die Objektgeometrie einer Kugel oder eines Würfels genutzt.
Eine andere Möglichkeit dieses Verfahren zu benutzen, ist die Spiegelung der Umgebung auf Objekten zu simulieren. Ziel ist die Nachbildung ideal spiegelnder Objekte. Das Objekt muss im Mittelpunkt des virtuellen Körpers sein und der Beobachterstandpunkt im Intervall von Mittelpunkt zu Innenseiten des virtuellen Körpers [21, S. 303]. Das Grundprinzip gleicht dem Raytracing Verfahren in ersten Schritten. Ein Strahl a wird vom Beobachterstandpunkt zum spiegelnden Objekt gesendet, trifft dabei auf die Facetten. Die Facettennormale n dient als Lot und aus dem entsprechenden Winkel wird ein Reflexionsvektor r zu den Texturkoordinaten der Umgebung ermittelt. Der Vorgang ist in Abbildung 6.4 verdeutlicht. Anschließend wird das Texel

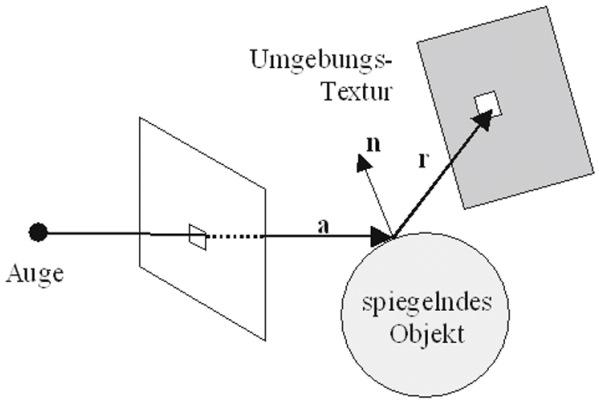


Abbildung 6.4.: Veranschaulichung des Environmental Mapping Verfahrens
[10, S. 328]

der Umgebungstextur auf die Facette des Objekts aufgebracht.
Das Verfahren funktioniert ebenfalls bei bewegten Objekten. Die Bedingung hierfür ist, das die Bewegung vom Objekt zur Umgebung sehr klein ist. Bei größeren Objektänderungen muss das Bild neu gerendert werden [10, S. 328 ff.]. Durch das starke Vereinfachen von Spiegelungen entstehen folgende Nachteile. Falls mehrere Objekte in der Szene sind, können sie sich nicht selbst reflektieren. Die Genauigkeit ist auf zwei Stufen beschränkt, entweder auf die Größe der Facetten oder der Pixel [23, S. 275 ff.].
In Abbildung A.22 ist ein Beispiel aus dem Ventuz Designer zu sehen. Dabei wurden sechs Texturen aus unterschiedlichen Blickrichtungen verwendet. Die Texturen wurden aus sechs Fotos mit 90° Öffnungswinkel gewonnen, welche auf der Innenseite eines Kubus aufgebracht sind. Die Bilder sind übergangslos miteinander verbunden. Im Zentrum befindet sich die spiegelnde Kugel. Es gibt die Möglichkeit die Texturen für die sechs Blickrichtungen vom Computer rendern zu lassen.

KAPITEL 7

Zusammenfassung

In der vorliegenden Arbeit wurden die mathematischen-physikalischen Grundlagen zur Abbildung der realen Welt vereinfacht beschrieben, um diese mit der Software Ventuz Designer zu simulieren. Auf Themengebiete, wie die Animationstechniken, interaktive und externe Steuerung und Datenschnittstellen, die der Ventuz Designer und Director zur Verfügung stellt, ist auf Grund des Umfangs dieser schriftlichen Arbeit nicht eingegangen worden. Die vorgestellten $Nodes$ dienen zur Manipulation des Renderprozesses, ohne die Kenntnis von Programmiersprachen vorauszusetzen. Das Verständnis der dahinter liegenden Verfahren ist geeignet, um dreidimensionale Szenen zu erstellen, zu gestalten und zu präsentieren. Diese Arbeit stellt eine einführende Literatur dar. Im ersten Teil wurden die Grundlagen eines Computers und der Computergrafik erklärt. Auf das Thema dreidimensionale Objekte und deren Darstellung im virtuellen Raum wurde eingegangen. Verfahren der Visualisierung, Beleuchtungsmodelle und die Nutzung von Texturen wurden beschrieben. Im Anhang befinden sich Bildschirmfotos vom Ventuz Designer.

Literatur

[1] Thomas Akenine-Möller, Eric Haines und Naty Hoffman. *Real-Time Rendering*. 3. Aufl. ISBN: 978-1-4398-6529-3. Natick: A.K. Peters Ltd., 2002.

[2] Günter Aumann und Klaus Spitzmüller. *Computerorientierte Geometrie*. 1. Aufl. Mannheim: BI-Wissenschaftsverlag, 1993.

[3] Michael Bender und Manfred Brill. *Computergrafik - Ein anwendungsorientiertes Lehrbuch*. 2. Aufl. ISBN: 3-446-40434-1. München: Carl Hanser Verlag, 2006.

[4] Gert Bär. *Geometrie: eine Einführung für Ingenieure und Naturwissenschaftler*. 2. Aufl. ISBN: 3-519-20722-2. Leipzig: Teubner, 2001.

[5] Axel Bruns. *Die Geschichte des Computer: Wie es bis zur Form des heutigen "PC"kam.* 1. Aufl. ISBN: 978-3-7380-2145-5. München: neobooks Self-Publishing, 2015.

[6] José Encarnação, Wolfgang Straßer und Reinhard Klein. *Graphische Datenverarbeitung 2 - Modellierung komplexer Objekte und photorealistische Bilderzeugung*. 4. Aufl. ISBN: 3-486-23469-2. München: Oldenbourg, 1997.

[7] Walter Eversheim. *100 Jahre Produktionstechnik - Werzeugmaschinenlabor WZL der RWTH Aachen von 1906-2006*. 1. Aufl. Berlin: Springer, 2006.

[8] Georg Glaeser. *Geometrie und ihre Anwendungen*. 2. Aufl. ISBN: 978-3-8274-1797-8. München: Spektrum, 2007.

[9] Walter-Dieter Klix. *Konstruktive Geometrie - darstellend und analytisch*. 1. Aufl. ISBN: 3-446-21566-2. München: Carl Hanser Verlag, 2001.

[10] Alfred Nischwitz u. a. *Computergrafik und Bildverarbeitung Band 1*. 3. Aufl. ISBN: 978-3-8348-1304-6. Wiesbaden: Vieweg+Teubner, 2011.

[11] *Quelle Abbildung installierter Großrechner RWTH Aachen*. 1.12.2015. URL: https://doc.itc.rwth-aachen.de/display/VE/HPC-Historie+im+RZ+der+RWTH.

[12] *Quelle Beispiel für verschiedene Perspektiven*. 4.01.2016. URL: http://www.arakanga.de/axonometrische_projektionen/.

[13] *Quelle Darstellungsverfahren der Schrift Calibri*. 11.12.2015. URL: http://www.microsoft.com/typography/otspec/TTCH01.htm.

[14] *Quelle Meilenstein 3dfx Voodoo Graphics*. 5.12.2015. URL: http://www.pcgameshardware.de/Grafikkarten-Grafikkarten-97980/News/3dfx-stellt-die-Voodoo-Graphics-vor-PCGH-Retro-Spezial-698841/.

[15] *Quelle Rechenzeit für ein Bild von Toy Story*. 3.12.2015. URL: http://collider.com/pixar-numbs-toy-story-brave/.

[16] *Quelle Technische Beschreibung Großrechner Zuse Z22*. 1.12.2015. URL: http://museum.informatik.uni-kl.de/Rechner/Zuse/Z22/Dokumente/Z22%20Technische%20Beschreibung.pdf.

[17] *Quelle Technische Daten Nvidia Quadro M4000*. 1.12.2015. URL: http://www.nvidea.de/object/quadro-desktop-gpu-specs-de.html.

[18] *Quelle Trend Integration CPU und GPU*. 5.12.2015. URL: http://www.amd.com/en-us/press-releases/Pages/amd-fusion-apu-era-2011jan04.aspx.

[19] *Quelle für Beschreibung von Software und Firma Ventuz*. 4.01.2016. URL: http://www.ventuz.com/.

[20] *Quelle menschliche Auge, Intergationszeit der Photorezeptoren*. 1.12.2015. URL: https://e3.physik.uni-dortmund.de/~suter/Vorlesung/Medizinphysik_12/5_Auge.pdf.

[21] Hans-Günther Schiele. *Computergrafik für Ingenieure*. 1. Aufl. ISBN: 978-3-642-23842-0. Berlin: Springer, 2012.

[22] Konrad Stich. *40 Jahre NC- und Antriebstechnik im Werkzeugmaschinenbau*. 1. Aufl. Berlin: Pro Buisness, 2013.

[23] Alan Watt. *3D-Computergrafik*. 3. Aufl. ISBN: 3-8273-7014-0. München: Pearson Studium, 2002.

ANHANG A
Abbildungen Ventuz Designer

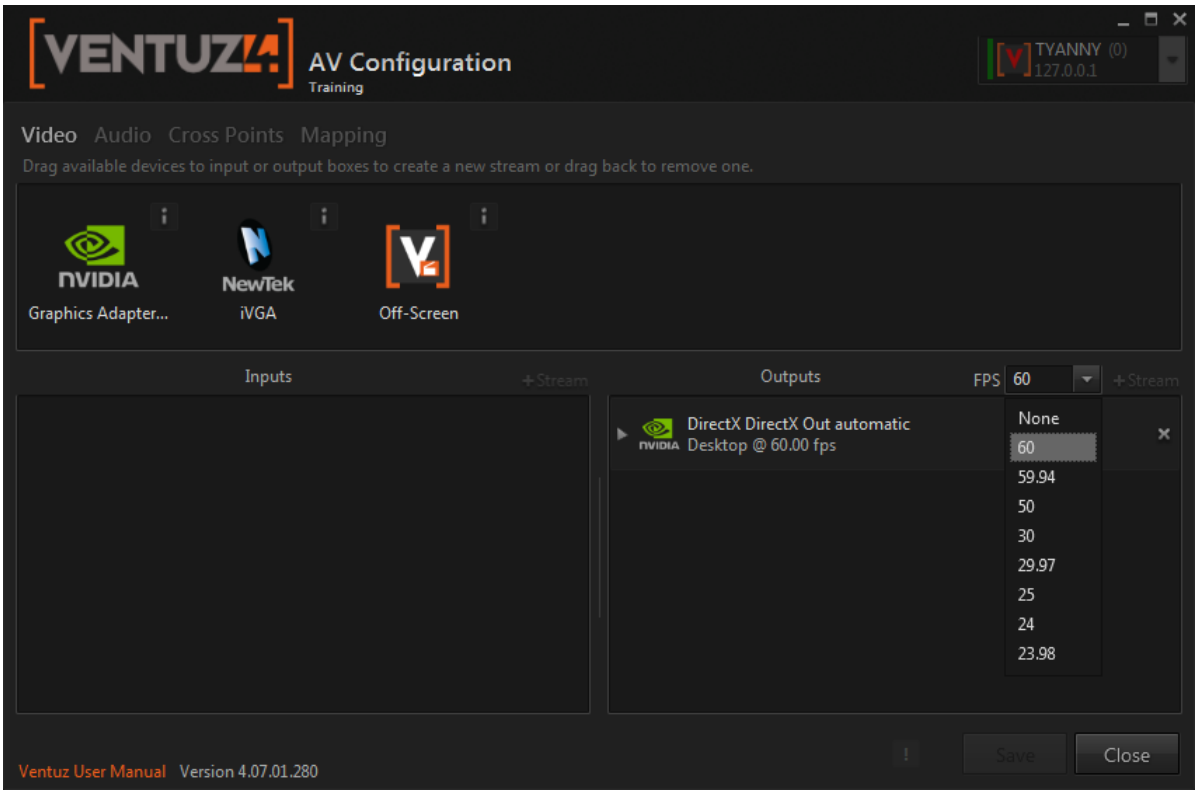


Abbildung A.1.: Einstellung der Bildwiederholrate

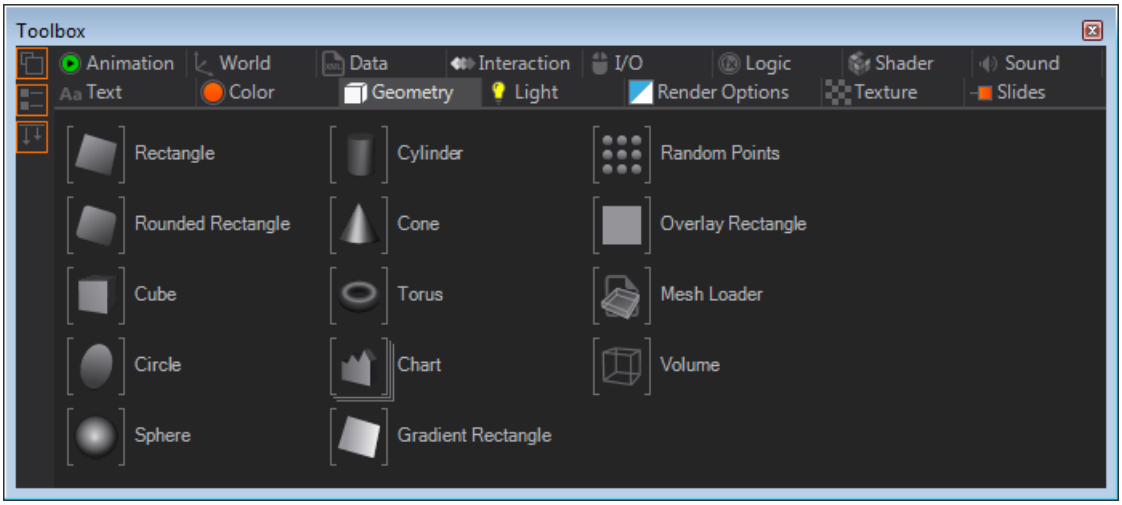


Abbildung A.2.: Verfügbare Geometrieprimitive

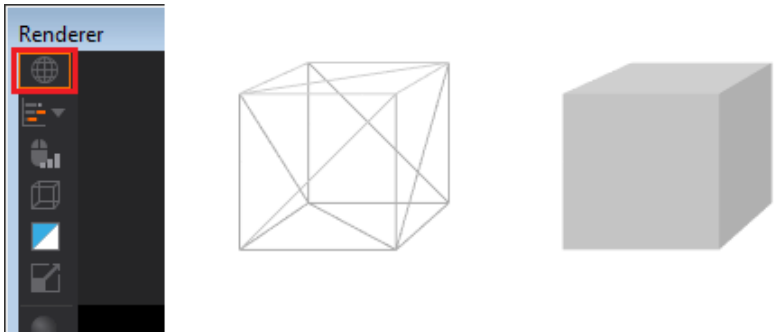


Abbildung A.3.: Umschaltung zwischen Drahtgittermodell und Grenzflächenmodell

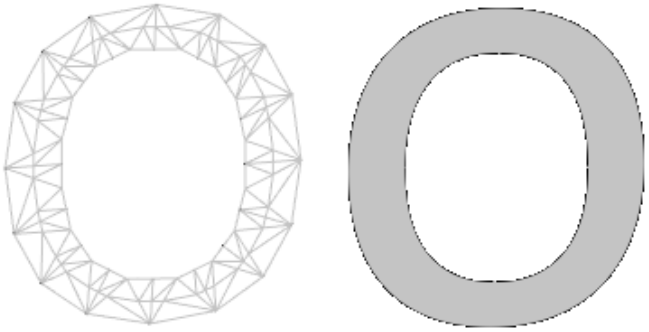


Abbildung A.4.: Beispiel eines Oberflächenmodells

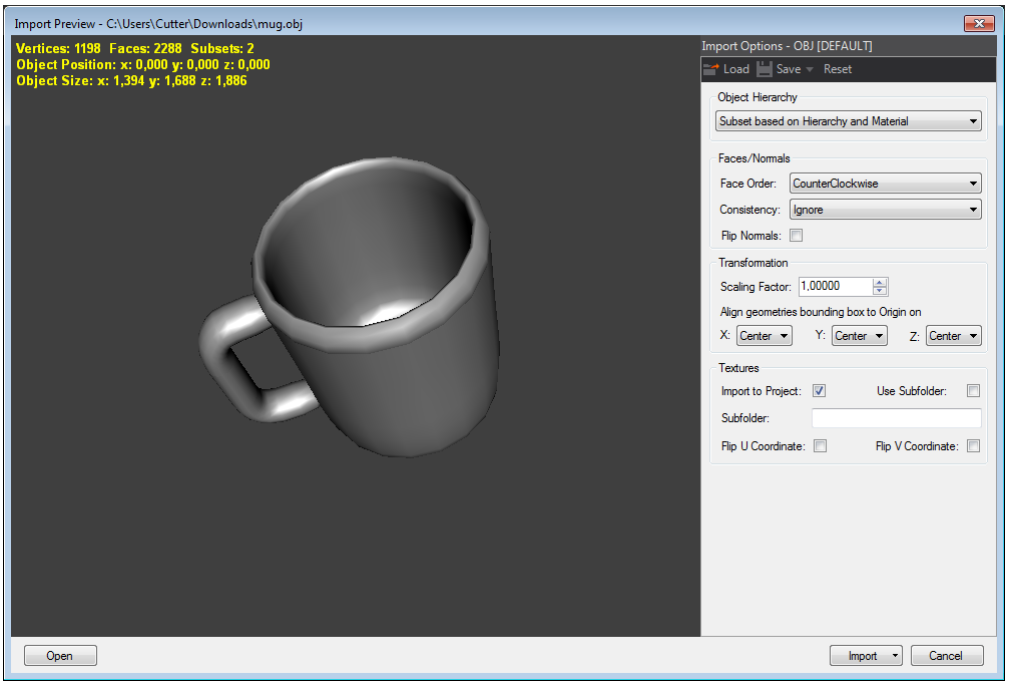


Abbildung A.5.: Import eines Modells

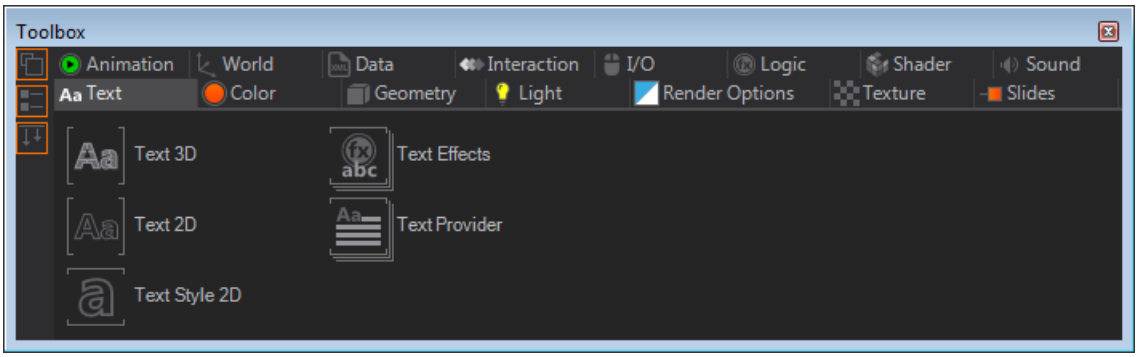


Abbildung A.6.: Verfügbare Textmodule

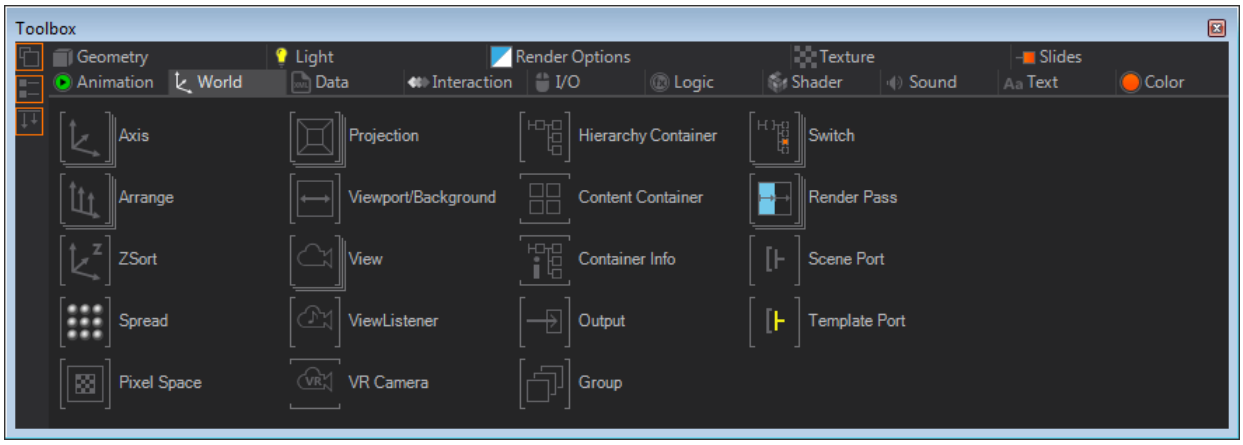


Abbildung A.7.: Verfügbare Optionen für den dreidimensionalen Raum

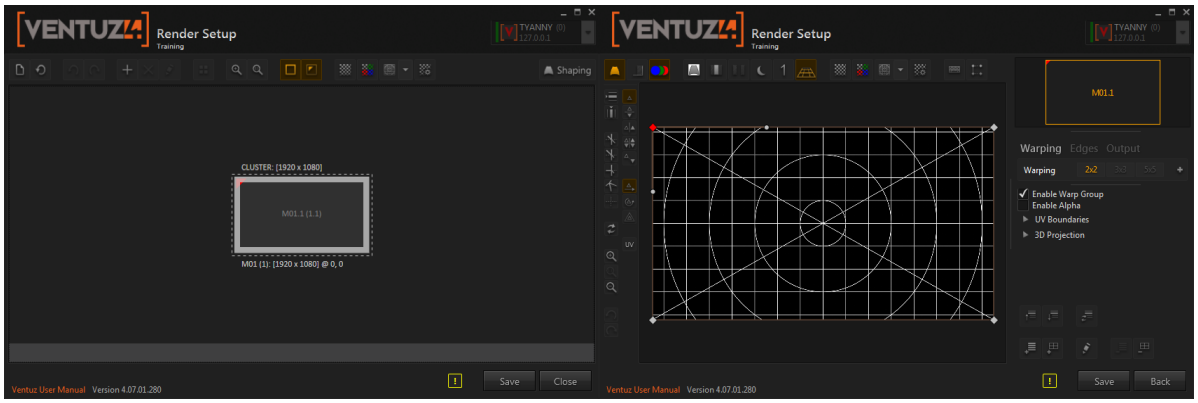


Abbildung A.8.: Einstellungen bei der Viewporttransformation

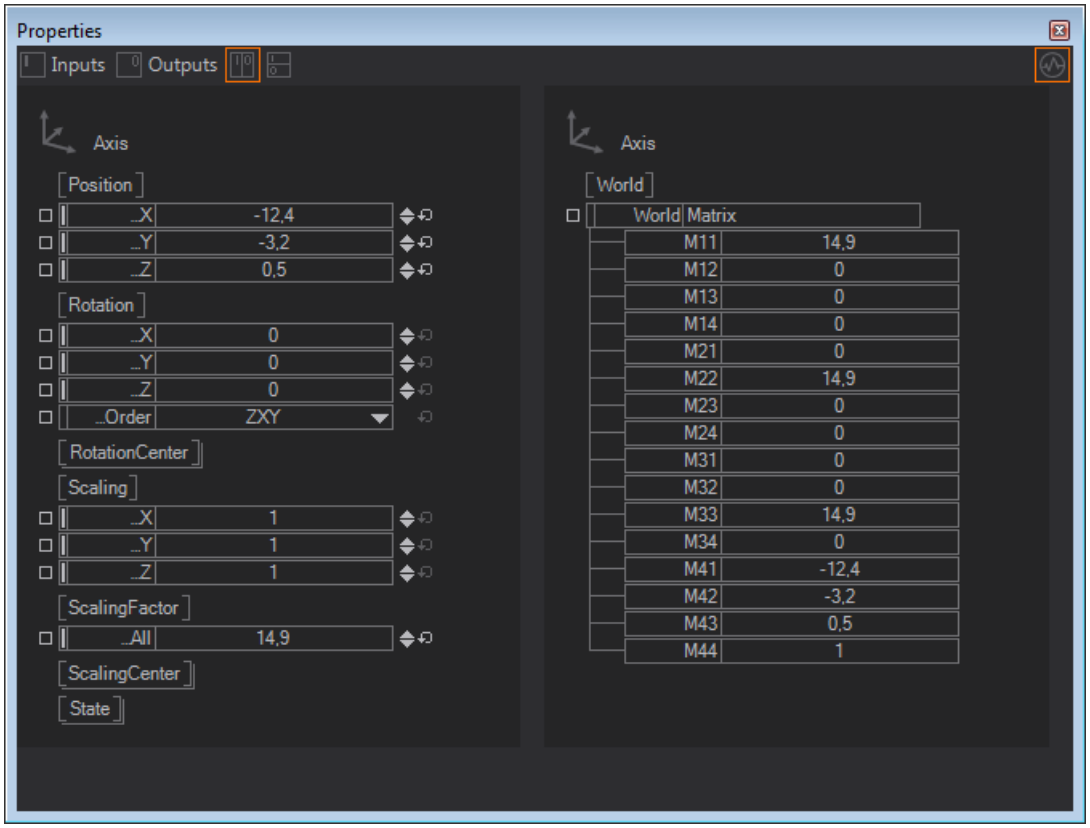


Abbildung A.9.: Axis Node mit Transformationsmatrix in Ventuz

Abbildungen Ventuz Designer

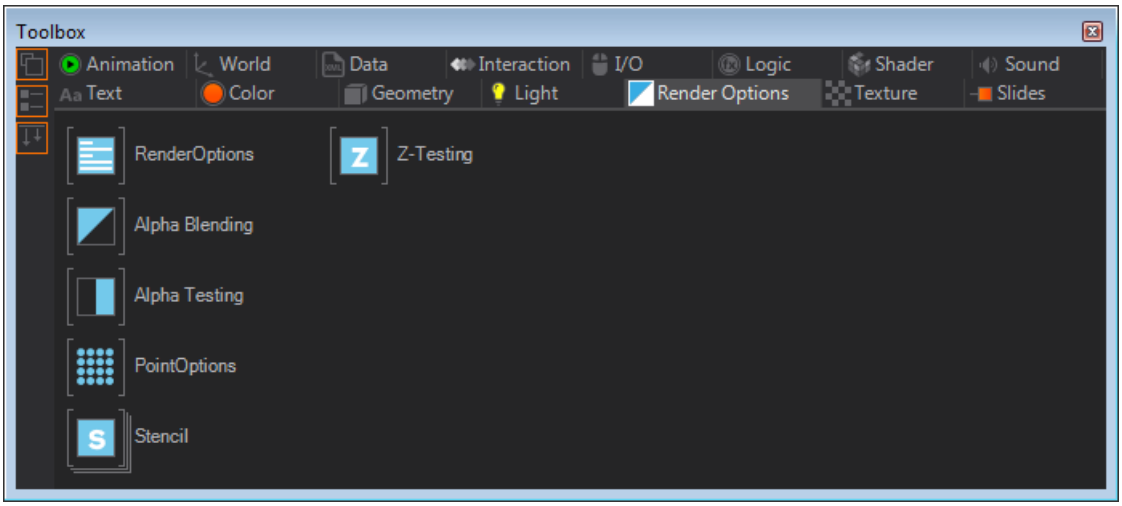


Abbildung A.10.: Verfügbare Optionen für die Visualisierung

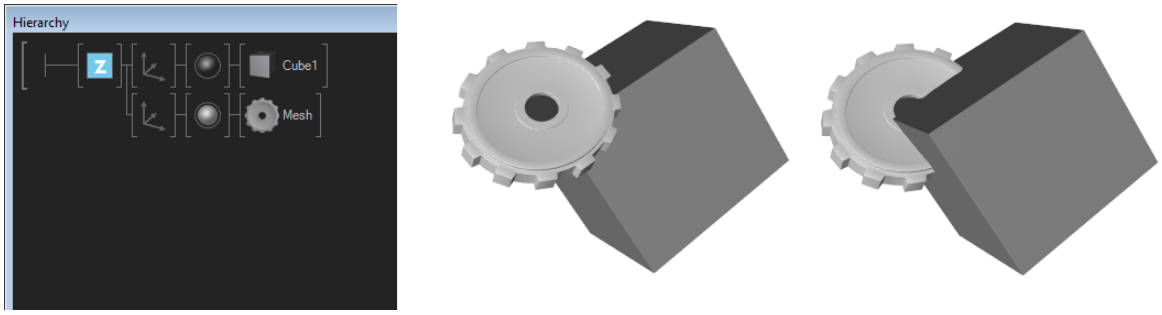


Abbildung A.11.: Beispiel Maleralgorithmus und Z-Buffer

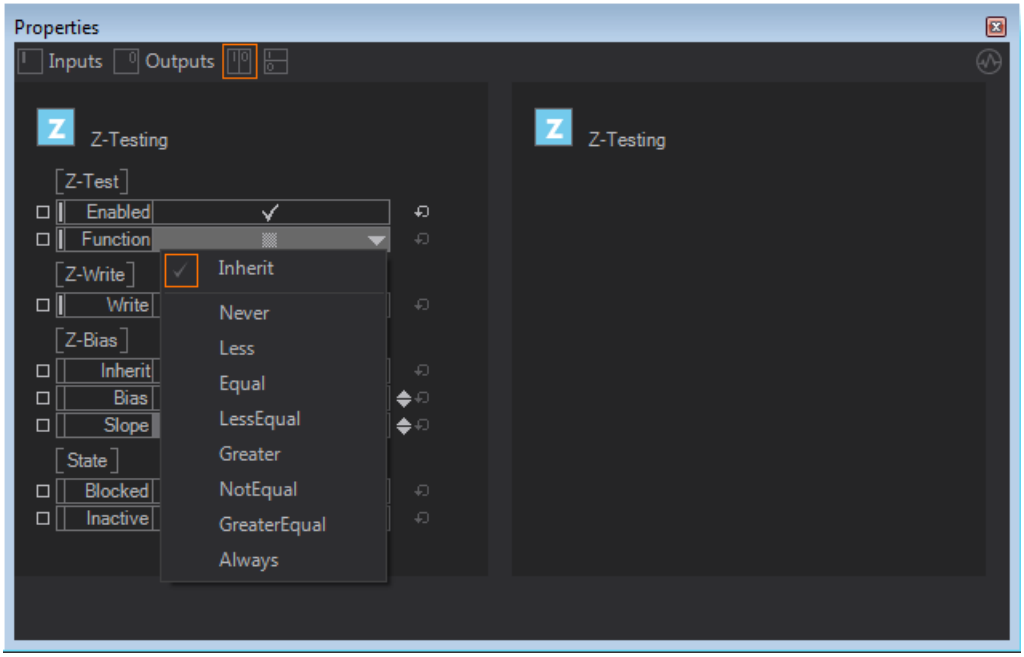


Abbildung A.12.: Verfügbare Optionen beim Z-Buffer Algorithmus

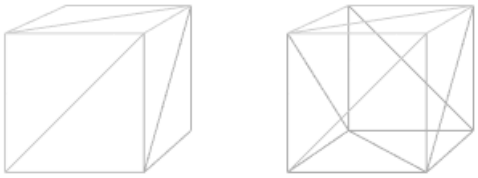


Abbildung A.13.: Kubus mit und ohne Rückseitenentfernung in der Drahtgitteransicht

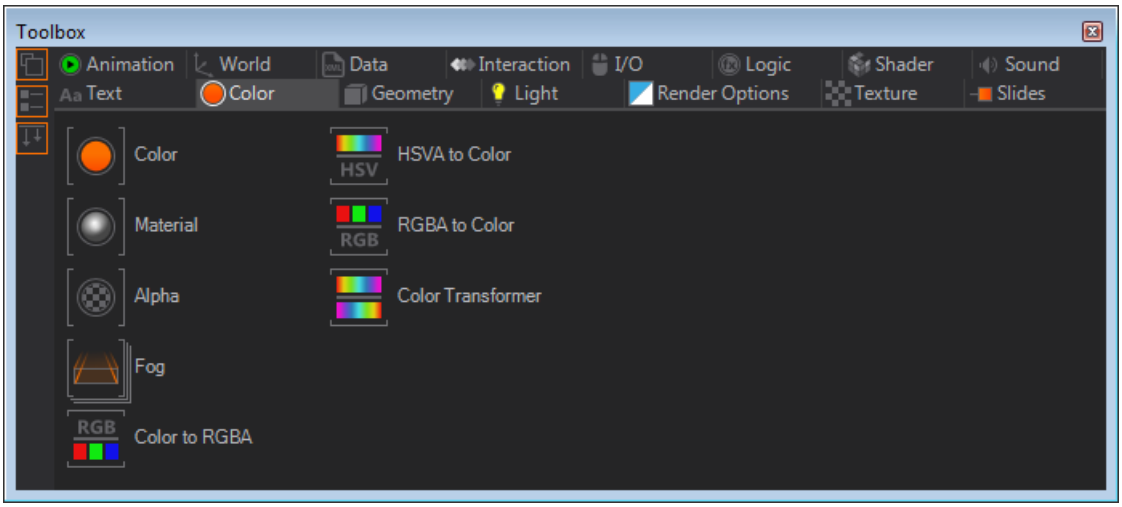


Abbildung A.14.: Verfügbare Optionen für die Farbe

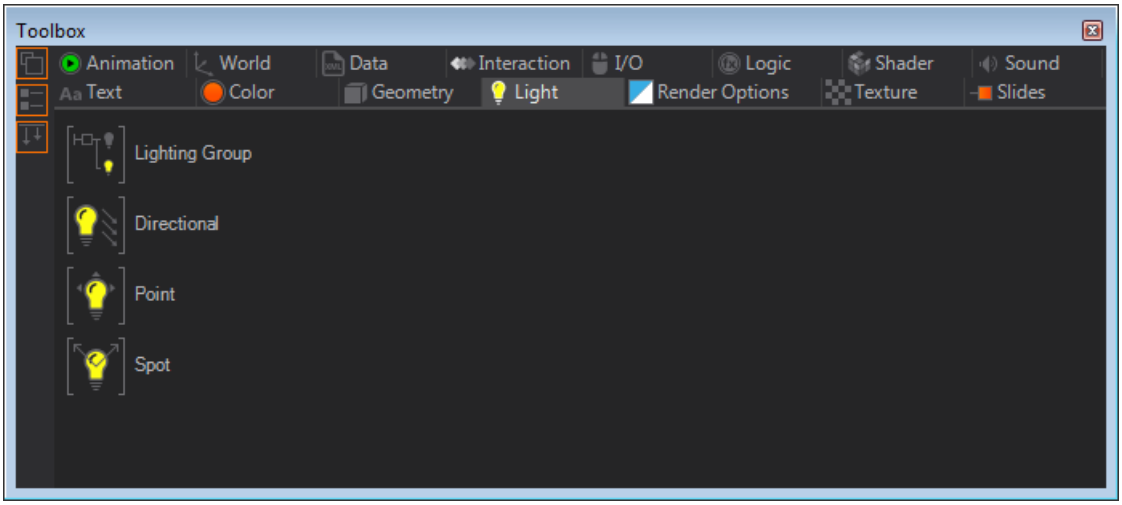


Abbildung A.15.: Verfügbare Lichtquellen

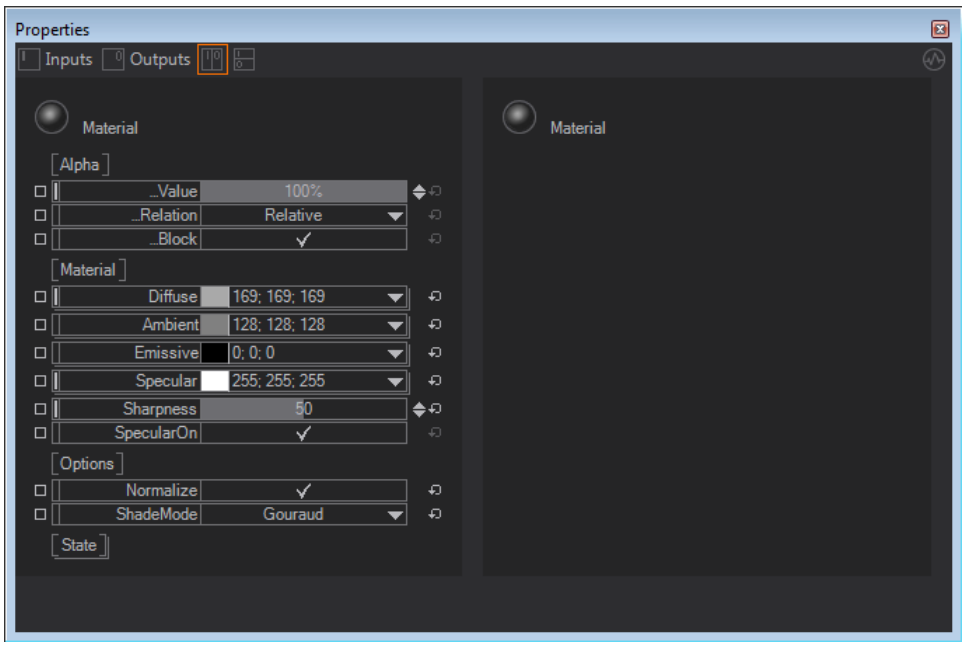


Abbildung A.16.: Einstellungen bei der Beleuchtungsberechnung

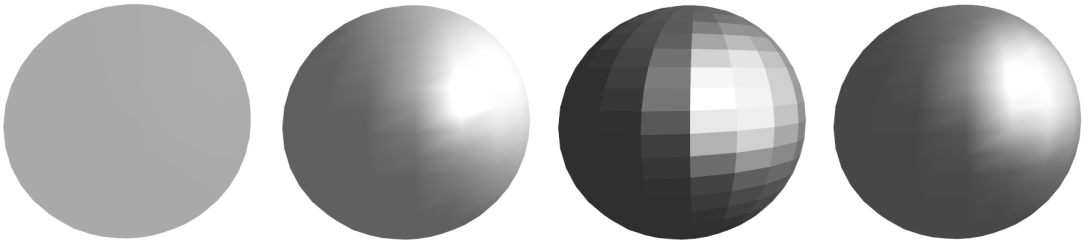


Abbildung A.17.: Beispiel für verschiedene Schattierverfahren

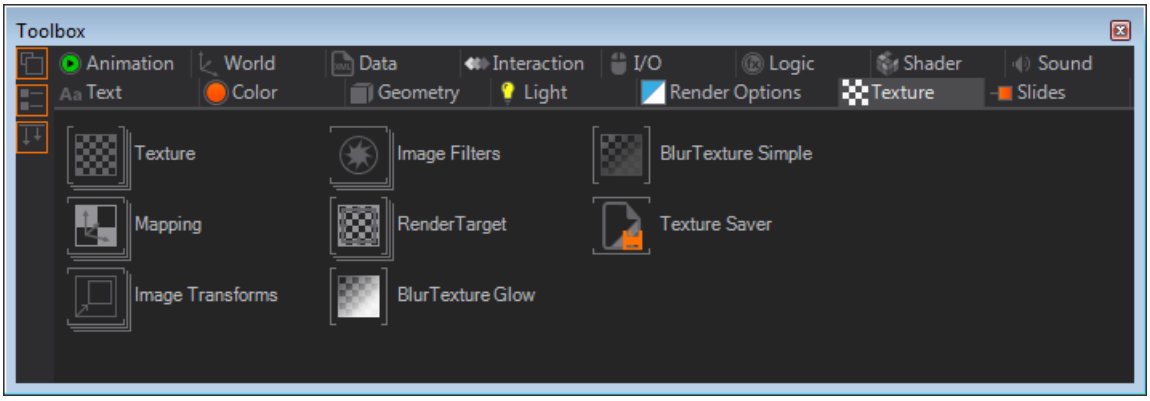


Abbildung A.18.: Verfügbare Optionen für die Texturierung

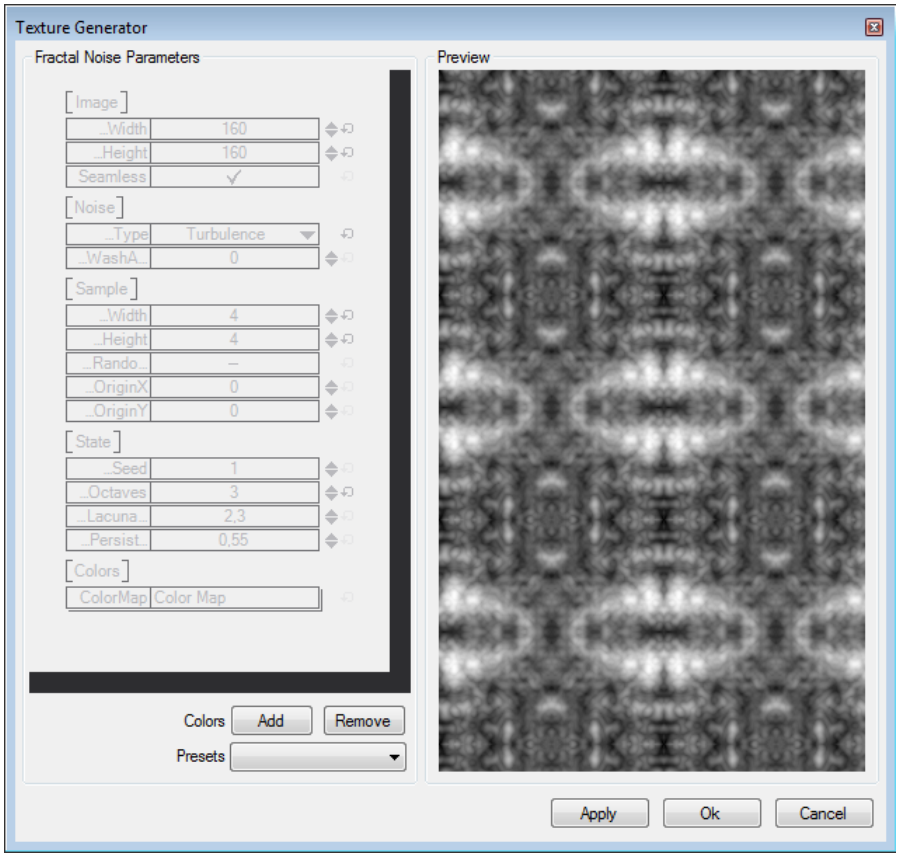


Abbildung A.19.: Erzeugung einer prozeduralen Textur

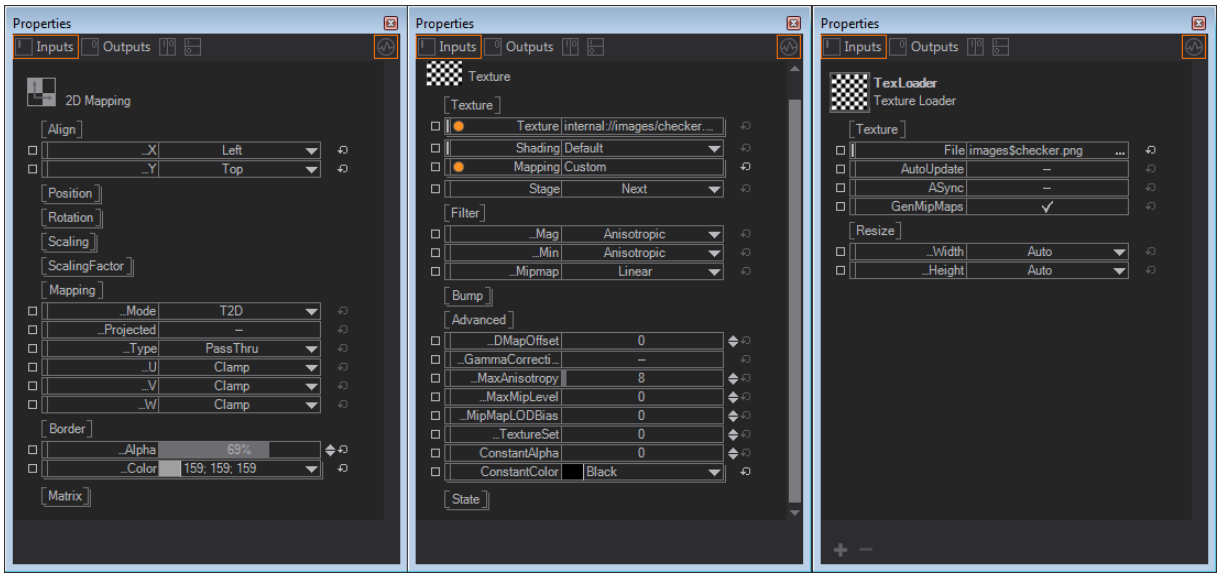


Abbildung A.20.: Verfügbare Mappingoptionen aus ausgewählten Nodes

Abbildung A.21.: Beispiel eines Bump-Mappings

Abbildung A.22.: Beispiel eines Enviromental-Mappings mit virtuellen Kubus

Selbständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig angefertigt und keine anderen als die von mir angegeben Hilfsmittel und Quellen verwendet habe. Insbesondere versichere ich, dass ich alle wörtlichen und sinngemäßen Übernahmen aus anderen Werken als solche kenntlich gemacht habe.

Ort, Datum, Name: