Kapitel 15

Viewing Pipeline

Die Abbildung dreidimensionaler Objekte auf dem Bildschirm wird in eine Reihe von Elementartransformationen zerlegt:

- Konstruktion von komplexen Szenen aus elementaren Objekten (*Modeling*),
- Festlegen der Bildebene (*View Orientation*),
- Projektion auf ein normiertes Gerät (*View Mapping*),
- Abbildung auf ein Ausgabegerät (*Device Mapping*).

15.1 Die synthetische Kamera

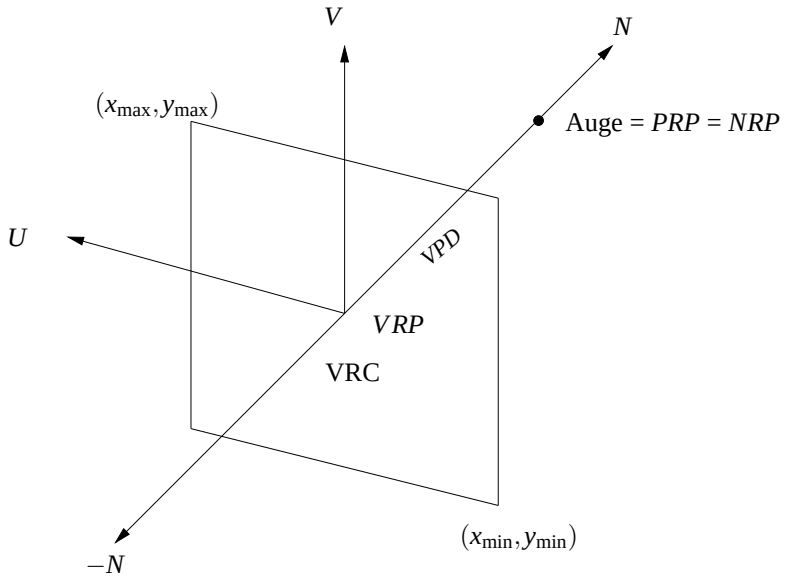

Abbildung 15.1: Parameter zur Beschreibung der synthetischen Kamera

Die Parameter zur Durchführung der 3D-auf-2D-Abbildung (*viewing transformation*) sind im wesentlichen die Position und Orientierung einer *synthetischen Kamera* und ein Punkt in der Szene, auf den die Kamera fokussiert ist: Der View Reference Point (*VRP*).

Die View Plane ist die Bildebene, auf die die Szene projiziert wird. Diese ist definiert durch die View Plane Normal N, die Richtung, aus der die Kamera die Szene aufnimmt, und den *VRP*. Die Kamera befindet sich im *PRP* (Projection Reference Point oder Normal Reference Point (*NRP*)), der dem Augenpunkt eines natürlichen Betrachters entspricht.

Die Orientierung der Kamera wird durch den View Up Vector (*VUV* oder *VUP*) festgelegt. Durch Projektion des *VUV* in die View Plane erhält man den Vektor V. Der Vektor U in der View Plane wird so gewählt, daß U, V, N ein rechtshändiges kartesisches Koordinatensystem bilden, das sogenannte View Reference Coordinate System VRC mit dem *VRP* als Ursprung und dem Augenpunkt *PRP* auf der positiven N-Achse bei $N = VPD$ (View Plane Distance).

Die beiden Punkte in den UV-Koordinaten der Viewplane $(x_{\min}, y_{\min})$, $(x_{\max}, y_{\max})$ legen das sogenannte View Window fest, d.h., was von der Szene zu sehen ist (entspricht der Brennweite eines Kameraobjektivs). Der sichtbare Teil kann noch weiter eingeschränkt werden durch Angabe einer *front plane* und einer *back plane*.

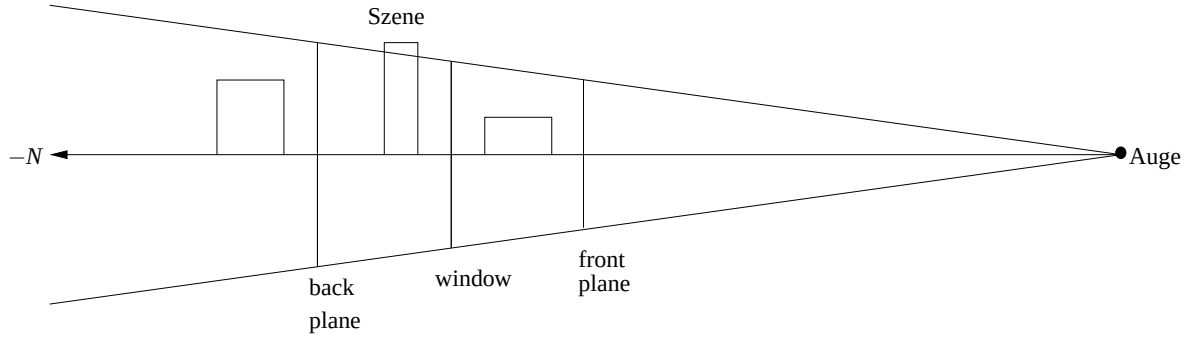

Abbildung 15.2: Einschränkung des sichtbaren Teils

Bemerkung: Zur Berechnung der Projektion auf die Bildebene ist es unerheblich, wo die Bildebene in der Szene arrangiert ist.

15.2 Viewing Pipeline

Die fotorealistische Darstellung von 3D-Objekten einer Szene auf den 2D-Bildschirm läßt sich beschreiben durch eine Sequenz von Transformationen, genannt *Viewing Pipeline*.

Folgende Informationen sind erforderlich:

- Jedes Objekt wird beschrieben durch Modellkoordinaten (z.B. Würfel ist beschrieben durch Mittelpunkt $(0,0,0)$ und Kantenlänge 1).
- Die Szene wird beschrieben durch eine Menge von Objekten, deren Lage und Größe in Weltkoordinaten beschrieben sind.

- Die Beleuchtung wird beschrieben durch eine Menge von Lichtquellen, der Lage und Ausrichtung in Weltkoordinaten beschrieben sind.
- Die synthetische Kamera wird beschrieben durch $U, V, N, VPD, (x_{\max}, y_{\max}), (x_{\min}, y_{\min})$.

Folgende Abkürzungen werden verwendet:

MC	model coordinate system
WC	world coordinate system
VRC	view reference coordinate system
NPC	normalized projection coordinate system
DC	device coordinate system

Jedes Polygon durchläuft die folgende Viewing-Pipeline:

1.) Modeling: MC → WC
Beschreibe Polygonpunkte in Weltkoordinaten (dies geschieht durch Translation, Rotation und Skalierung).

2.) View Orientation: WC → VRC
Beschreibe Polygonpunkte bzgl. des UVN-Systems (dies geschieht durch Wechsel des Koordinatensystems, anschließend ist die Szene beschrieben mit xy-Ebene = Bildebene, das Auge liegt bei $z = VPD$).

3.) View Mapping: VRC → NPC
Transformiere die Szene derart, daß der sichtbare (= im Pyramidenstumpf aus view window, front plane, back plane liegende) Teil abgebildet wird auf einen Einheitswürfel, dessen Vorder- und Rückseite der *front plane* bzw. *back plane* entsprechen.

4.) Device Mapping: NPC → DC
Bilde jeden Bildpunkt auf Gerätekoordinaten ab = übernimm x, y-Koordinaten; z liefert Tiefeninformation.

15.2.1 Modeling-Transformationen

Punkt 1.) wird *Modeling* genannt: Das Anordnen von Objekten aus dem *Modellkoordinatensystem* (MC) zu einer Szene in dem sogenannten *Weltkoordinatensystem* (WC). Im MC liegen die Objekte als Prototypen vor. Ihre Definitionspunkte sind unabhängig von der späteren Größe und Position im WC in Modellkoordinaten gegeben. Der Ursprung des MC befindet sich sinnvollerweise im Zentrum des Objekts. Erst durch das Modeling erhalten die Objekte im WC ihre individuelle Größe, Orientierung und Position. Dieses wird im wesentlichen durch drei Arten von Transformationen realisiert: Translation, Rotation und Skalierung.

15.2.2 View Orientation

Zur Abbildung einer dreidimensionalen Szene aus den Weltkoordinaten auf den Bildschirm muß eine natürliche Betrachtersicht (*View Orientation*) definiert werden. Eine solche Betrachtersicht besteht aus den Viewing-Parametern

- Betrachterstandpunkt *PRP* (*Projection Reference Point*; auch *Eyepoint*),
- Blickrichtung *VRP* (*View Reference Point*),
- vertikale Orientierung *VUV* (*View Up Vector*) oder *VUP* (*View Up Point*),
- Blickwinkel (Brennweite).

Der fiktive Betrachter blickt vom *PRP* in Richtung *VRP* und kippt dabei die Kamera so um die Blickrichtung, daß *VUP* von ihm aus gesehen vertikal nach oben zeigt. Der Blickwinkel entspricht der Brennweite und regelt den Bildausschnitt (*Zooming*).

Durch diese Parameter wird das VRC (*View Reference Coordinate System*) definiert mit *VRP* als Ursprung und *PRP* auf der positiven *z*-Achse. Die *xy*-Ebene wird als *View Plane* bezeichnet und steht senkrecht auf der Blickrichtung. Die *y*-Achse ist gegeben durch die Projektion von *VUP* in die View Plane. Deshalb darf *VUP* nicht parallel zur Blickrichtung sein.

Das VRC hat die gleiche Metrik wie das WC. Es handelt sich also um einen Wechsel des Koordinatensystems. Die entsprechende Matrix, die den Wechsel vom WC ins VRC durchführt ergibt sich, wenn man die Vektoren *U*, *V* und *N* und die homogenen Koordinaten des *VRP* bzgl. des WC als Spaltenvektoren nebeneinander schreibt und diese Matrix anschließend invertiert.

Der Abstand zwischen *VRP* und *PRP* wird als *VPD* (*View Plane Distance*) bezeichnet.

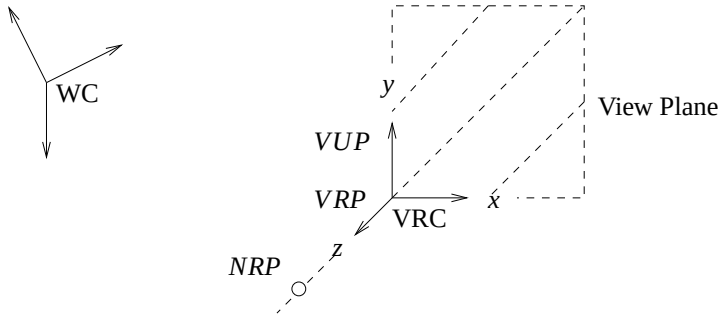

Abbildung 15.3: Definition des VRC

Für die Anwendung des Viewings ergeben sich im Programm verschiedene Möglichkeiten:

- Soll ein Objekt von verschiedenen Seiten betrachtet werden, so wird *VRP* in das Objekt gelegt. Durch *PRP* läßt sich nun die Blickrichtung frei wählen. Der Vektor *VUP* kann dabei konstant bleiben, sofern nicht senkrecht von oben oder unten geschaut wird.
- Soll sich die Szene bei konstantem Standpunkt um den Betrachter drehen, wird *PRP* festgesetzt. Durch Veränderung des *VRP* bewegt sich dann die Szene vor dem Auge des Betrachters. Der *VUP* kann dabei konstant bleiben, solange er nicht kollinear zum Vektor durch *VRP* und *PRP* wird.
- Eine weitere Möglichkeit sind Animationen, bei denen sich z.B. der Beobachter vorwärts in die Szene hineinbewegt. Dieser Effekt läßt sich durch Verschieben von *VRP* und *PRP* um denselben Vektor erzielen.

15.2.3 View Volume

Bei einer Kamera ist der maximal abbildbare Ausschnitt einer Szene durch das Objektiv festgelegt. Im Programm wird zur Erzielung desselben Effekts ein rechteckiger Ausschnitt aus der Bildebene — das *Window* — gewählt. Das Seitenverhältnis entspricht dem des Ausgabegeräts. Durch das View Window und die gewählte Projektion wird ein Bildraum definiert, der *View Volume* genannt wird. Nur jene Objekte und Objektteile, die sich innerhalb dieses Bildraumes befinden, werden auf die Bildebene (und dadurch in das View Window) abgebildet.

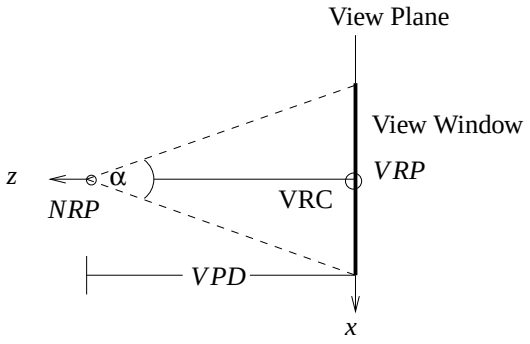

Abbildung 15.4: Definition des View Windows

Die Form des Bildraumes ist bei der perspektivischen Projektion eine unendliche Pyramide, deren Spitze im Projektionszentrum *PRP* (*Projection Reference Point*) liegt und deren Kanten durch die Eckpunkte des View Windows verlaufen. Hierbei wird im Programm der Betrachterstandpunkt *PRP* als Projektionszentrum gewählt.

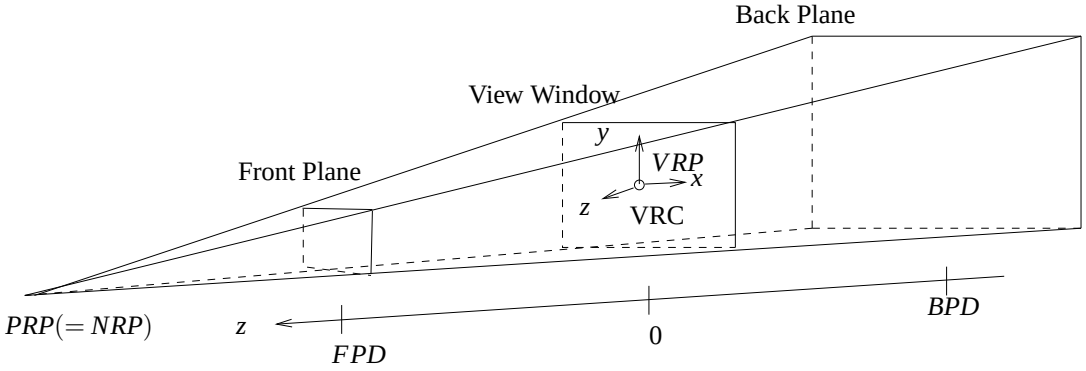

Abbildung 15.5: Bildraum bei perspektivischer Projektion

Die Begrenzung des Bildraumes in Richtung der *z*-Achse des VRC erfolgt durch zwei zur View Plane parallele Ebenen, die *Front* und *Back Plane* genannt werden. Die Front Plane liegt vom Projektionszentrum PRP aus gesehen vor der Back Plane. Ihre *z*-Komponenten werden als *Front Plane Distance* (*FPD*) und *Back Plane Distance* (*BPD*) bezeichnet. Es gilt $BPD < FPD$. Mit diesen Ebenen können Teile der Szene von der Projektion auf die Bildebene ausgeschlossen werden. Speziell bei der perspektivischen Projektion erweist sich diese Möglichkeit als notwendig, da sonst sehr nahe Objekte

alle anderen verdecken bzw. sehr weit entfernte als nicht mehr erkennbare (d.h. zu kleine) Figuren abgebildet würden. Mit Front und Back Plane ergibt sich als View Volume ein Pyramidenstumpf.

15.2.4 View Mapping

Statt nun das View Volume aus dem VRC direkt auf den Bildschirm zu projizieren, wird im Programm eine weitere Transformation durchgeführt, die nicht nur zur Effizienzsteigerung des Algorithmus führt, sondern auch die Projektion der Szene auf die Bildebene erleichert: Der Bildraum wird in einen zur Bildebene normal ausgerichteten Einheitswürfel $(0 \leq x \leq 1; 0 \leq y \leq 1; 0 \leq z \leq 1)$ umgewandelt. Anstelle mehrerer unterschiedlicher Projektionen muß so anschließend nur noch die orthogonale Parallelprojektion auf die Ebene $z = 0$ durchgeführt werden. Das sogenannte *View Mapping* wird deshalb auch als Ausgabe auf einen normierten Bildschirm bezeichnet. Das Koordinatensystem, in dem sich der Einheitswürfel befindet, heißt *Normalized Projection Coordinate System* (NPC). Der Betrachterstandpunkt befindet sich im Unendlichen auf der positiven z-Achse und hat die homogenen Koordinaten $(0,0,1,0)$. Vom Betrachter aus gesehen liegt im NPC der Punkt $(0,0,0)$ "links unten hinten" und $(1,1,1)$ "rechts oben vorne".

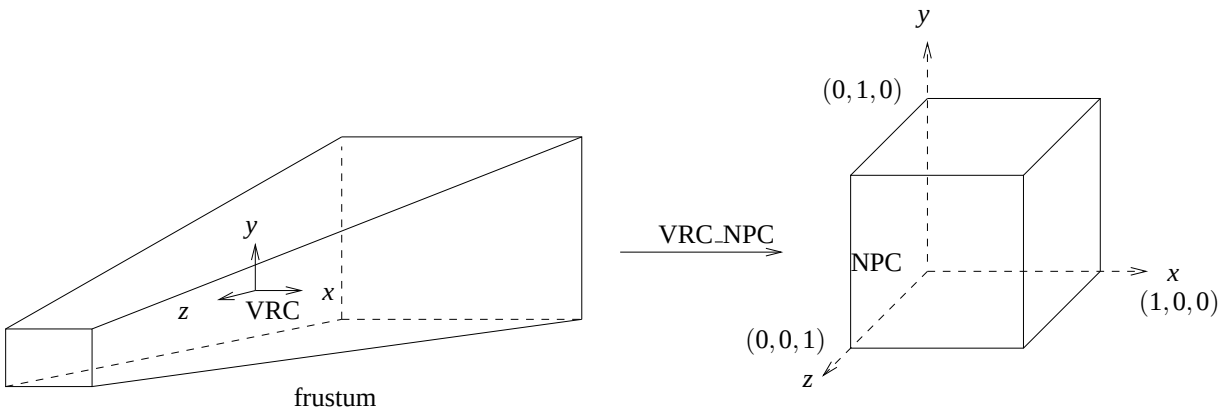

Abbildung 15.6: View Mapping

Bei der perspektivischen Projektion wird in einem ersten Schritt der Pyramidenstumpf, der den Bildraum darstellt und *frustum* genannt wird, auf einen regelmäßigen Pyramidenstumpf abgebildet. Dessen Grundfläche ist ein Quadrat, das mit jeder Seitenfläche einen Winkel von 45° einschließt. Dabei werden die z-Koordinaten nicht und die x- und y-Koordinaten proportional zur z-Koordinate verändert, was einer Scherung an der z-Achse entspricht. Im zweiten Schritt wird der regelmäßige Pyramidenstumpf in den normierten Einheitswürfel transformiert. Dazu müssen die x- und y-Koordinaten proportional zu den reziproken z-Werten skaliert werden. Die Front Plane entspricht im NPC der Ebene $z = 1$ und die Back Plane der Ebene $z = 0$.

Beide Schritte werden im Programm zu einer Transformationsmatrix zusammengefaßt. Bei deren Anwendung auf die homogenen Koordinaten ist zu beachten, daß die resultierenden Punkte im allgemeinen w-Werte ungleich 1 haben, die affinen Koordinaten also erst nach der Division durch w vorliegen. Außerdem ist diese Abbildung nur bezüglich der x- und y-Koordinaten linear. In z-Richtung werden die Werte durch die Scherung im zweiten Schritt reziprok verzerrt, d.h., äquidistante Punkte längs der z-Achse im VRC häufen sich im NPC bei $z = 0$ nahe der Back Plane.

Zur Erleichterung der Herleitung wird zunächst das Koordinatensystem so transformiert, daß der *PRP* im Ursprung sitzt. Danach wird das Koordinatensystem an der *xy*-Ebene gespiegelt, indem die *z*-Koordinaten mit -1 multipliziert werden. Danach ist das Koordinatensystem linkshändig. Es seien d_{min}, d und d_{max} die Abstände der Frontplane, Bildebene und Backplane vom Augenpunkt.

Zur Durchführung von Punkt 3.) der Viewing Pipeline wird zunächst die abgeschnittene Pyramide (= frustum) transformiert in einen symmetrischen Pyramidenstumpf mit quadratischer Grundfläche und Kanten unter 45°.

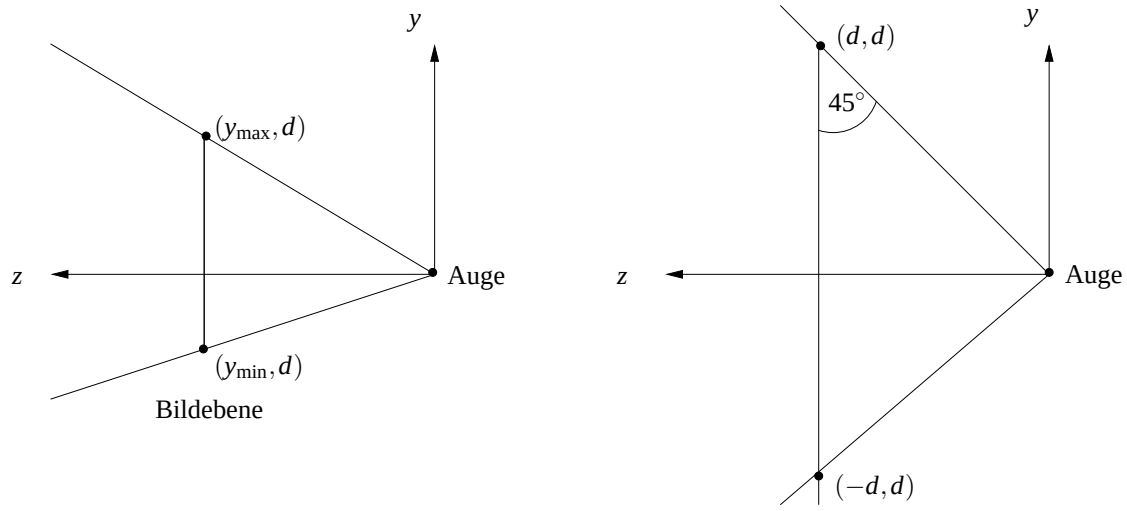

Abbildung 15.7: Überführung in Pyramidenstumpf

Es müssen also die *y*-Koordinaten proportional zur *z*-Koordinate verändert werden. Dies entspricht einer Scherung an der *z*-Achse, so daß die Achse vom *PRP* zum Zentrum des ViewWindows mti der *z*-Achse zusammenfällt. Zusätzlich wird in *y*-Richtung so skaliert, daß die Grundseite des entstandenen Pyramidenstumpfs eine Kantenlänge von $2d$ bekommt:

$$\begin{aligned} y' &= k_1 \cdot y + k_2 \cdot z \\ z' &= z \end{aligned}$$

Punkt $(y_{\max}, d)$ soll abgebildet werden auf (d, d);
Punkt $(y_{\min}, d)$ soll abgebildet werden auf $(-d, d)$.

Durch Lösen des Gleichungssystems

$$\begin{aligned} d &= k_1 \cdot y_{\max} + k_2 \cdot d \\ -d &= k_1 \cdot y_{\min} + k_2 \cdot d \end{aligned}$$

erhält man

$$k_1 = \frac{2d}{y_{\max} - y_{\min}}, \quad k_2 = -\frac{y_{\max} + y_{\min}}{y_{\max} - y_{\min}}$$

Analoge Überlegungen für die *x*-Werte ergibt:

$$k_1 = \frac{2d}{x_{\max} - x_{min}}, \quad k_2 = -\frac{x_{\max} + x_{\min}}{x_{max} - x_{min}}$$

Als nächstes wird die regelmäßige Pyramide in den Einheitswürfel $(0 \leq x \leq 1, 0 \leq y \leq 1, 0 \leq z \leq 1)$ transformiert. Die *front plane* entspricht der Ebene $z = 0$ und die *back plane* der Ebene $z = 1$.

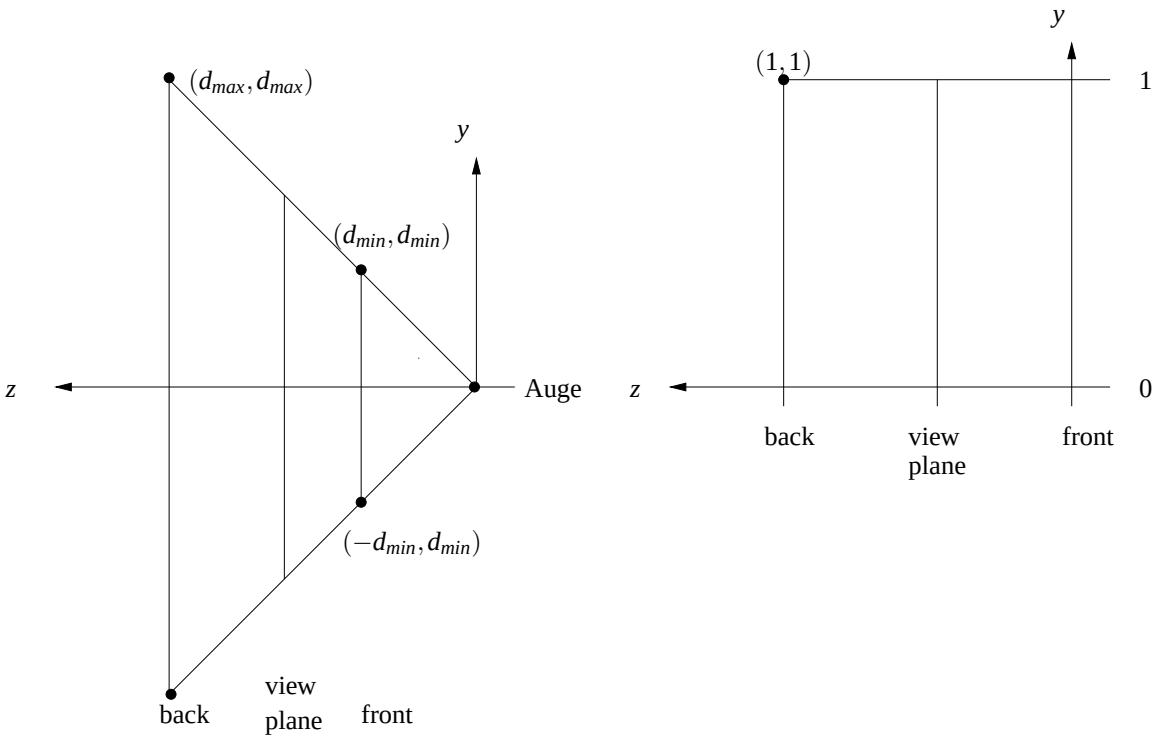

Abbildung 15.8: Überführung in Einheitswürfel

Da die y-Werte proportional zu den reziproken z-Werten skaliert werden müssen, ergibt sich als Transformation

$$\begin{aligned} y' &= k_1 + k_2 \cdot \frac{y}{z} \\ z' &= k_3 + k_4 \cdot \frac{1}{z} \end{aligned}$$

Der Kehrwert von z im Term zu y' ist nur möglich, indem durch einen geeigneten Eintrag in der vierten Zeile der noch zu konstruierenden Transformationsmatrix erreicht wird, dass die vierte Komponente des transformierten Punktes den Wert z enthält. Bei der üblichen Auswertung einer homogenen Koordinate wird dann durch z geteilt. Durch diesen Trick müssen aber neben dem Term für y' und dem für x' auch der Term zu z' den Kehrwert von z eingebaut bekommen.

Punkt $(d_{\max}, d_{\max})$ soll abgebildet werden auf $(1,1)$,
Punkt $(-d_{\min}, d_{\min})$ soll abgebildet werden auf $(0,0)$.
Durch Lösen des Gleichungssystems

$$1 = k_1 + k_2 \cdot \frac{d_{\max}}{d_{\max}}$$

$$0 = k_1 + k_2 \cdot \frac{-d_{\min}}{d_{\min}}$$

erhält man

$$k_1 = \frac{1}{2}, \quad k_2 = \frac{1}{2}.$$

Durch Lösen des Gleichungssystems

$$\begin{aligned} 1 &= k_3 + k_4 \cdot \frac{1}{d_{\max}} \\ 0 &= k_3 + k_4 \cdot \frac{1}{d_{\min}} \end{aligned}$$

erhält man

$$k_3 = \frac{d_{\max}}{d_{\max} - d_{\min}}, \quad k_4 = -\frac{d_{\min} \cdot d_{\max}}{d_{\max} - d_{\min}}.$$

Die x-Werte werden analog zu den y-Werten skaliert. Insgesamt ergibt sich somit

$$\begin{aligned} x' &= \frac{1}{2} + \frac{x}{2} \cdot \frac{1}{z} \\ y' &= \frac{1}{2} + \frac{y}{2} \cdot \frac{1}{z} \\ z' &= \frac{d_{\max}}{d_{\max} - d_{\min}} - \frac{d_{\min} \cdot d_{\max}}{d_{\max} - d_{\min}} \cdot \frac{1}{z}. \end{aligned}$$

Zwar wird hierdurch die Szene im vorderen z-Bereich nicht-linear gestaucht, zur Bestimmung der Sichtbarkeit reichen die ermittelten z-Werte jedoch aus, da ihre Ordnung erhalten bleibt.

Durch Verknüpfen der beiden letzten Transformationen erhält man in Schritt 3.) als Transformationsmatrix

$$\begin{bmatrix} \frac{d}{x_{\max} - x_{\min}} & 0 & \frac{1}{2}(1 - \frac{x_{\min} + x_{\max}}{x_{\max} - x_{\min}}) & 0 \\ 0 & \frac{d}{y_{\max} - y_{\min}} & \frac{1}{2}(1 - \frac{y_{\min} + y_{\max}}{y_{\max} - y_{\min}}) & 0 \\ 0 & 0 & \frac{d_{\max}}{d_{\max} - d_{\min}} & \frac{-d_{\max} \cdot d_{\min}}{d_{\max} - d_{\min}} \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

die den sichtbaren Teil der Szene in den Einheitswürfel transformiert.

Um wieder ein rechtshändiges Koordinatensystem zu erhalten, wird zunächst das Koordinatensystem so transformiert, daß die Back Plane in die xy-Ebene verschoben wird. Abschließend wird wieder an der xy-Ebene gespiegelt.

15.2.5 Device Mapping

Die abschließende Projektion der Szene in Schritt 4.) aus dem NPC auf den Bildschirm wird als *Device Mapping* bezeichnet. Der Einheitswürfel enthält dank der vorangegangenen Transformationen die gesamte darzustellende Szeneninformation. Die Abbildung muß lediglich die *x*- und *y*-Koordinaten aus dem NPC so in die Bildschirmkoordinaten DC (*Device Coordinate System*) transformieren, daß eine anschließende Rundung die ganzzahligen Koordinaten der Pixel ergibt.

DC ist auf den meisten Bildschirmen ein linkshändiges Koordinatensystem. (Die *y*-Achse zeigt nach unten, oben links ist der Ursprung $(0,0)$.) Die Anzahl der Pixel im Ausgabefenster ist flexibel und betrage horizontal *xsize* und vertikal *ysize*. In *x*-Richtung muß dann das Intervall $[0,1]$ auf die diskreten Werte $\{0,...,xsize-1\}$ und in *y*-Richtung $[0,1]$ umgekehrt auf $\{ysize-1,...,0\}$ abgebildet werden. Die *z*-Koordinaten dienen später zur Bestimmung und Unterdrückung verdeckter Flächen, die sich durch die Staffelung der Objekte in der Tiefe des Bildraumes ergeben.

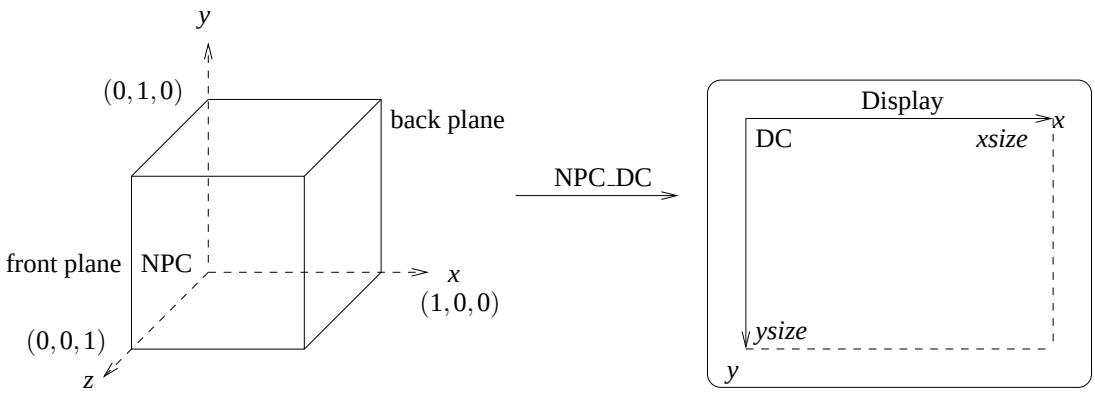

Abbildung 15.9: Device Mapping

Die Transformationsmatrix entspricht einer Skalierung um den Vektor $(xsize,-ysize,1)$ konkateniert mit einer Translation des Ursprungs in die linke untere Ecke des Bildschirms $(0,ysize,0)$.

$$T_{\text{NPC\_DC}} = \begin{bmatrix} xsize & 0 & 0 & 0 \\ 0 & -ysize & 0 & ysize \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Für die Projektion werden einfach die *x*- und *y*-Werte übernommen; die *z*-Werte sind erforderlich zur Regelung der Sichtbarkeit.

15.2.6 Zusammenfassung

Abbildung 15.10 zeigt den Ablauf der Transformationen im Überblick. Diese *Viewing Pipeline* wird von den Eckpunkten aller Polygone durchlaufen, aus denen sich die Szene zusammensetzt.

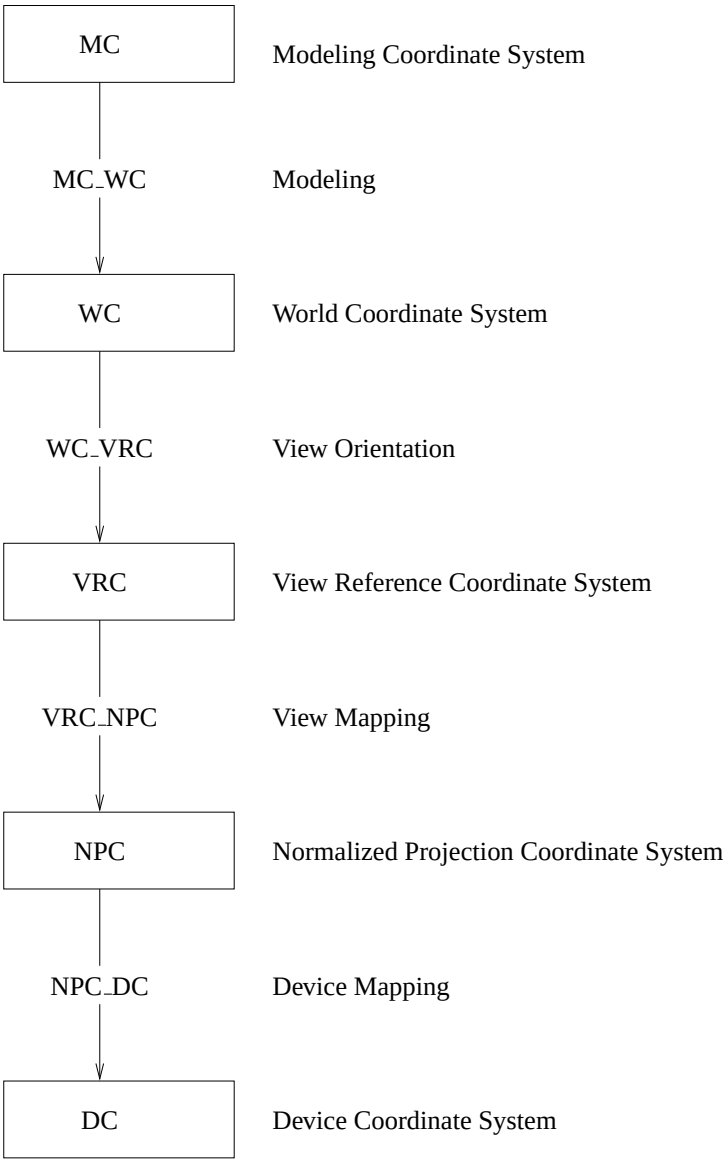

Abbildung 15.10: Transformationspipeline

15.3 Clipping

Jedes Polygon, das die Viewing Pipeline durchläuft, kostet Rechenaufwand zur Transformation seiner Eckpunte und damit Zeit. Noch mehr Zeit kostet das Schattieren des Polygons, das wir später noch kennenlernen werden.

Da nur der Teil der Szene, der sich im *frustum* befindet, später auf dem Bilbschirm zu sehen sein wird, ist es sinnvoll, den unsichtbaren Teil der Szene so früh wie möglich im Ablauf der Viewing Pipeline "loszuwerden". Im average case dürfen die Punkte als gleichverteilt im Raum angenommen werden. Der unsichtbare Teil der Szene wird also im Allgemeinen einen nicht zu vernachlässigenden Anteil an der Gesamtszene haben.

Um den unsichtbaren Teil auszublenden, müssen wir die gesamte Szene auf irgendeiner Stufe der Viewing Pipeline am *frustum* clippen. Wir wollen zwei Stufen, die sich anbieten, näher betrachten:

- Clipping im WC
- Clipping im NPC

Zunächst gehen wir davon aus, daß neben den MC-Koordinaten auch die WC-Koordinaten für jeden Polygonpunkt vorliegen.

15.3.1 Clipping im WC

Diese Strategie beginnt damit, die Ebenengleichungen für die sechs Seiten des *frustums* im WC zu bestimmen. Bei einer Kamerafahrt muß dies im Allgemeinen für jedes Frame neu geschehen. Der Aufwand ist aber unabhängig von der Szene und ihrer Komplexität. Jeder Polygonpunkt wird durch Auswertung der Ebenengleichung an jeder der sechs Seiten geclippt. Wenn ein Polygon eine oder mehrere der Seiten schneidet, müssen noch die neuen Polygonpunkte berechnet werden, indem die Schnittpunkte der Polygonkanten mit den Seiten ermittelt werden.
Nur die verbleibenden Punkte müssen ins DC transformiert und auf dem Bildschirm angezeigt werden. Dazu ist eine Multiplikation mit der Matrix WC_DC notwendig.

15.3.2 Clipping im NPC

Zunächst müssen alle Polygonpunkte mit der Matrix WC_NPC ins NPC transformiert werden. Dann wird die dreidimensionale Erweiterung des 2D-Polygon-Clippings von Cohen und Sutherland durchgeführt. Es wird für jeden Punkt ein 6-Bit-Bereichscode bestimmt, der angibt, ob ein Punkt im View Volume (der Einheitswürfel) liegt oder nicht. Dieser Code läßt sich besonders einfach bestimmen, da die Clippingebenen unabhängig von der Szene und der Lage der Kamera immer die Seiten des Einheitswürfels sind. Es reicht also ein einfacher Vergleich mit den Werten 0 bzw. 1, um ein Bit des Bereichscodes festzulegen. Falls eine Polygonkante eine oder mehrere der Seiten schneidet müssen wieder die neuen Polygonpunkte errechnet werden.
Danach müssen die verbleibenden Punkte mit einer weiteren Multiplikation (mit NPC_DC) in Bildschirmkoordinaten gebracht und angezeigt werden.

15.3.3 Vergleich der beiden Vorgehensweisen

Beim Clipping im WC spart man eine Transformation **aller** Punkte im Gegensatz zum Clipping im NPC. Dafür ist das eigentliche Clipping eines Polygon(punkte)s etwas aufwändiger als im zweiten Fall. Auch die Schnittpunktberechnung ist im NPC etwas günstiger als im WC.
Beim Clipping im WC muß das *frustum* noch ins WC gebracht werden. Dieser Vorgang hat konstanten Aufwand und fällt ab einer gewissen Szenengröße nicht mehr ins Gewicht.
Die Zahl der Punkte, die nach dem Clipping übrig bleibt ist in beiden Fällen gleich, damit auch der Aufwand für die abschließende Transformation.

Man kann nicht entscheiden, welche Strategie die bessere ist, denn die Effizienz der mathematischen Operationen auf dem verwendeten Prozessor bzw. in der verwendeten Programmiersprache spielen eine Rolle.

Der unnötige Aufwand der durch die Transformation der unsichtbaren Punkte ins NPC entsteht, läßt sich folgendermaßen abschätzen:

Angenommen die ganze Szene passe in einen Würfel der Kantenlänge g. Dann hat sie ein Volumen von g^3. Der Betrachter stehe im Schwerpunkt des Würfels und blicke so in die Szene, daß das View Window genau einer Würfelseite entspricht. Dann sieht er fünf Sechstel der Szene nicht. Bei gleichverteilten Punkten entspricht das auch fünf Sechstel unnötig vom WC ins NPC transformierten Punkte.

Jede Transformation verursacht 16 Multiplikationen und 12 Additionen. Die Entscheidung, ob ein Punkt im Frustum liegt, kostet im NPC sechs Vergleiche (die Hälfte der Vergleiche findet gegen 0 statt, was vermutlich nochmal schneller als ein beliebiger Vergleich ist). Dieselbe Entscheidung kostet im WC 18 Multiplikationen und 6 Vergleiche. D.h. das Clipping im WC ist pro Knoten zwei Mutliplikationen teurer. Allerding braucht man im NPC zusätzlichen Speicherplatz für die NPC-Koordinaten und den Bereichscode.

Wenn die WC-Koordinaten allerdings nicht vorliegen, sondern für jedes Frame neu aus den MC-Koordinaten errechnet werden müssen, sieht es etwas anders aus. Dann bleibt der Aufwand des Clipping im NPC gleich, denn der erste Schritt besteht jetzt in einer Transformation mit MC_NPC. Der Aufwand des Clipping im WC steigt aber um eine Transformation MC_WC für jeden Polygonpunkt.

Trotzdem hat die parallele Speicherung der WC-Koordinaten ihre Berechtigung. Sie werden für die Beleuchtung benötigt (hier allerdings nur die der sichtbaren Punkte) und viele Datenstrukturen, die die Bestimmung des sichtbaren Teils der Szene wesentlich effizienter machen, arbeiten auf den WC-Koordinaten.

15.3.4 Umgebungsclipping

Eine weitere Effizienzsteigerung wird möglich durch die Verwendung eines Umgebungsclippings. Hier wird ein Cluster von mehreren, komplexen Objekten mit einem großen Quader umgeben. Ergibt ein erster Clipping-Test, dass dieser Quader außerhalb des Frustums liegt, so erübrigen sich alle Clipping-Abfragen bzgl. seiner inneren Objekte.