

Radiometrische Beleuchtungssimulation mit präziser Sensormodellierung

Christoph Riesinger

25. April 2008

Zusammenfassung

Diese Arbeit beschäftigt sich mit der Erstellung einer Software im Zuge eines Programmierpraktikums namens *Zybesi* (**Z**ylinder**b**eleuchtung**s**imulator). Sie soll am Computer realistische Bilder von Zylindern unter Verwendung eines Beleuchtungsmodells generieren. Es handelt sich also um ein Simulationstool. Dabei wird zunächst auf die theoretischen und mathematischen Grundlagen, wie die Modellierung der Beleuchtung und der virtuellen Kamera eingegangen. Anschließend werden einige Aspekte der Implementierung genauer vorgestellt. Das Ende dieser Arbeit bildet neben einigen Anmerkungen zur fertigen Software Zybesi eine Aufzählung von erwähnenswerten Erkenntnissen, die sich während des Programmierpraktikums heraus kristallisiert haben.

1 Einleitung und Motivation

Eine moderne Form der Vermessung und Qualitätskontrolle von industriell gefertigten Bauteilen, beispielsweise für den Automobil- oder Maschinenbau, ist das digitale Fotografieren dieser Bauteile und die anschließende digitale Analyse durch einen Computer unter Verwendung diverser Algorithmen der Bildverarbeitung. Auf diese Weise lassen sich unter bestimmten Voraussetzungen und Annahmen, beispielsweise Form- und Lageparameter des Bauteils, äußerst exakt bestimmen.

Jedoch stecken derartige Ansätze noch in den Kinderschuhen und es besteht ein hoher Forschungsbedarf. Ein möglicher Ansatz zum Lösen des obigen Problems sieht wie folgt aus: Man unterteilt das zu vermessende Bauteil in seine geometrischen Primitiven wie Quader, Kegel, Tori oder Zylinder. Anstatt nun die Form- und Lageparameterbestimmung am komplexen Bauteil vorzunehmen, wird dies mit den geometrischen Primitiven gemacht. Zu diesem Zweck wird der Quader, Kegel, Tori oder Zylinder am Computer simuliert und virtuelle Bilder von ihm erstellt. Dabei wird so lange an den Form- und Lageparametern des virtuellen Objekts variiert, bis das durch Simulation entstandene Bild dem in der Realität aufgenommenen Bild möglichst genau entspricht. Um eine derartige Simulation durchzuführen bedarf es allerdings der realitätsnahen Modellierung der Szene, zu der das Primitiv (siehe Abschnitt 2.2), die Kamera (siehe Abschnitt 2.3) und ein Beleuchtungsmodell, das der tatsächlichen Beleuchtung möglichst gut entspricht (siehe Abschnitt 2.4), gehören. Bei diesem Programmierpraktikum geht es um die Erstellung der Software Zybesi, die eben eine solche realitätsnahe Simulation mit Zylindern durchführt. In den

nächsten Abschnitten werden zunächst die theoretischen und mathematischen Modelle erläutert, die verwendet wurden, um solche Simulationsbilder zu generieren (siehe Abschnitt 2). Anschließend wird auf das Vorgehen bei der Implementierung eingegangen (siehe Abschnitt 3) und zum Schluss werden neben der Software selbst (siehe Abschnitt 5) diverse Erkenntnisse und Erfahrungen mit den verwendeten Modellen präsentiert (siehe Abschnitt 4).

Inhalt dieses Programmierpraktikums ist lediglich die Simulation der Beleuchtung von Zylindern. Die oben in Form der Motivation beschriebene Vermessung von tatsächlichen Zylindern unter Verwendung der Simulationsbilder ist nicht Teil des Programmierpraktikums, da zu diesem Zweck noch weitere, hier nicht erwähnte Zwischenschritte notwendig sind.

2 Theoretische Grundlagen

In diesem Kapitel werden die theoretischen Grundlagen und mathematischen Modelle erläutert, die es Zybese ermöglichen sollen, eine realistische Beleuchtung eines Zylinders zu simulieren. Darüber hinaus werden im folgenden Abschnitt diverse Konventionen festgelegt. Die praktische Umsetzung dieser Grundlagen wird dann in den Kapiteln 3 und 5 beschrieben.

2.1 Modellierung des Koordinatensystems

Die Modellierung des Zylinders findet in einem kartesischen Koordinatensystem $(c, (u_1, u_2, u_3))$ statt, wobei die erste Koordinatenachse auch als x -Achse, die zweite als y -Achse und die dritte als z -Achse bezeichnet wird. Dabei ist c der Ursprung des Koordinatensystems. u_1 und u_2 spannen die xy -Ebene auf, die parallel zur Retinaebene der Kamera ist, die in 2.3 näher erläutert wird.

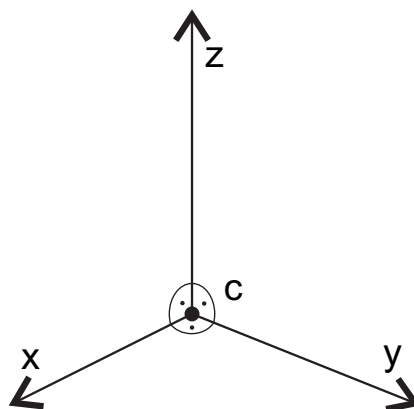


Abbildung 1: Kartesisches Koordinatensystem $(c, (u_1, u_2, u_3))$ im Raum mit x -, y - und z -Achse und optischen zentrum $c = (0, 0, 0)$.

2.2 Modellierung des Zylinders

Das von Zybesi verwendete geometrische Primitiv, der Zylinder, lässt sich mit fünf Parametern vollständig modellieren. Vier davon beschreiben die Lage des Zylinders im Raum, einer beschreibt die Form des Zylinders. Bei den vier Lageparametern handelt es sich einerseits um zwei Winkel α_1 und α_2 , die die Rotation der Zylinderhauptachse um die x - und z -Achse des Weltkoordinatensystems angeben und andererseits um zwei Werte d_1 und d_2 , die die Verschiebung des Durchstoßpunktes d der Zylinderhauptachse durch die xy -Ebene in x - und y -Richtung festlegen. Der einzig nötige Formparameter ist der Radius r des Zylinders.

Zu Beginn geht man davon aus, dass die Richtung der Zylinderhauptachse mit der y -Achse zusammen fällt, also dem Vektor $u_2 = (0, 1, 0)$ entspricht. Als anfänglicher Durchstoßpunkt der Zylinderhauptachse durch die xy -Ebene wird der Ursprung $c = (0, 0, 0)$ angenommen. Nun erfolgt zunächst eine Rotation der Zylinderhauptachse mit den Winkel α_1 um die x -Achse. Die Drehrichtung entspricht dabei dem Uhrzeigersinn, wenn man vom Ursprung c in Richtung der x -Achse u_1 blickt. Anschließend erfolgt eine Rotation der bereits einmal rotierten Zylinderhauptachse mit den Winkel α_2 um die z -Achse. Die Drehrichtung entspricht dabei dem Uhrzeigersinn, wenn man vom Ursprung c in Richtung der z -Achse u_3 blickt. Auf diese Weise legt man die Richtung f der Zylinderhauptachse fest. Als nächster Schritt erfolgt eine Translation der Zylinderhauptachse in x -Richtung um den Wert d_1 und in y -Richtung um den Wert d_2 . Der Punkt $(d_1, d_2, 0)$ gibt also den Durchstoßpunkt der Zylinderhauptachse durch die xy -Ebene an.

Auf diese Weise lässt sich jeder Zylinder beschreiben mit Ausnahme derjenigen, deren Hauptachse parallel zur xy -Ebene liegen und die somit keinen Durchstoßpunkt d durch die xy -Ebene besitzen.

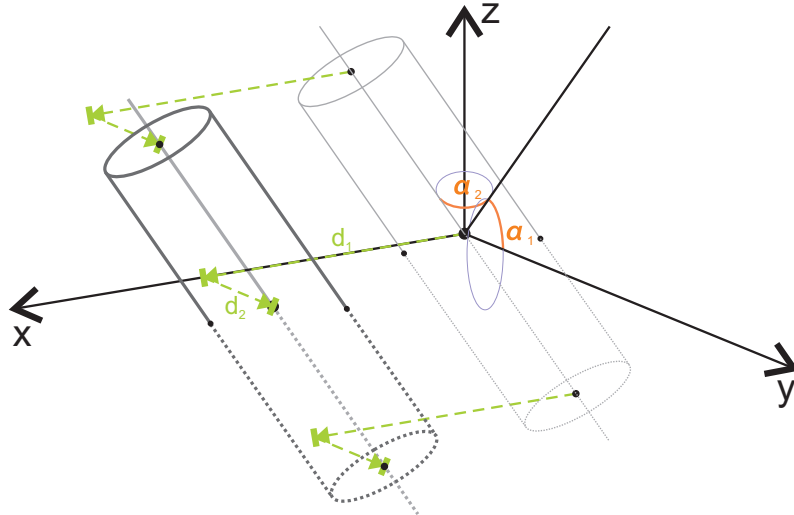


Abbildung 2: Modellierung des Zylinders mit Rotationswinkel α_1 und α_2 (orange) und xy -Ebenendurchstoßpunkt $d = (d_1, d_2, 0)$ (grün).

2.3 Modellierung der Kamera

Die virtuelle Kamera, die die simulierte Szene aufnimmt, wird als Lochkamera modelliert. Ihr optisches Zentrum c ist der Ursprung des Weltkoordinatensystems $(0, 0, 0)$. Die Retinaebene ist die $z = -1$ -Ebene, deren Ursprung der Punkt $(0, 0, -1)$ ist, also die orthogonale Projektion des Ursprungs des Weltkoordinatensystems c in die Retinaebene der Lochkamera. Somit ergibt sich das Kamerakoordinatensystem aus $((0, 0, -1), (u_1, u_2))$.

Um eine Rastergrafik zu erhalten, ist der lichtempfindliche Bereich einer Digitalkamera in Sensoren unterteilt, die matrixförmig angeordnet das Sensorarray bilden. Jeder einzelne Sensor korrespondiert zu einem Bildpunkt der Rastergrafik. Zybesi modelliert quadratische Sensoren, die den kompletten lichtempfindlichen Bereich der virtuellen Lochkamera abdecken. Die Ränder eines Sensors sind also automatisch auch die Ränder der Nachbarsensoren. Zwischen den Sensoren existiert bei Zybesi kein Abstand.

Die Ausdehnung des lichtempfindlichen Bereichs der virtuellen Lochkamera in der Breite (x -Richtung) beträgt genau 1. Die Höhe (y -Richtung) des lichtempfindlichen Bereichs steht dabei im selben Verhältnis zur Breite, wie die vertikale Auflösung des zu erstellenden Bildes zur horizontalen Auflösung. Auf diese Weise wird sicher gestellt, dass die Bildpunkte der virtuellen Kamera quadratisch sind, also die selbe Breite wie Höhe besitzen.

Der rechte Rand des lichtempfindlichen Bereichs wird als x_{max} bezeichnet und ist in x -Richtung genau so weit vom Ursprung der Retinaebene entfernt wie der linke Rand x_{min} . Somit ist $x_{min} = -x_{max}$. Analog verhält es sich mit dem oberen und unteren Rand des lichtempfindlichen Bereichs y_{max} und y_{min} : $y_{min} = -y_{max}$. Die Funktion der Zentralprojektion, die sog. Lochkamerabbildung, lautet

$$h : \mathbb{R}^3 \mapsto \mathbb{R}^3 : (x, y, z) \mapsto \left(-\frac{x}{z}, -\frac{y}{z}, -1\right)$$

bzw.

$$h : \mathbb{R}^3 \mapsto \mathbb{R}^2 : (x, y, z) \mapsto \left(-\frac{x}{z}, -\frac{y}{z}\right).$$

Als $b = (b_1, b_2, -1)$ werden die Projektionspunkte in der Retinaebene bezeichnet. Zusammen mit dem optischen Zentrum c legt b dann die Richtung eines Sichtstrahls $s = c - b$ fest.

2.4 Beleuchtungsmodell

Als Beleuchtungsmodell wird in der Computergrafik allgemein ein Verfahren bezeichnet, das die Reflexion von Lichtquellen an Oberflächen simuliert. Zybesi verwendet ein lokales Beleuchtungsmodell, welches eine Vorschrift zur Berechnung der Lichtintensität an einem bestimmten Oberflächenpunkt ist. Man unterscheidet allgemein zwischen empirischen und analytischen Modellen: Empirische Modelle sind schnell berechenbar, bauen aber auf keinerlei physikalischer Grundlage auf. Im Gegensatz dazu beschreiben die analytischen Modelle das physikalische Verhalten von Lichtreflexion, sind dafür aber wesentlich komplexer in ihrer Berechnung. In dieser Arbeit wird das Beleuchtungsmodell von Blinn verwendet. Zu beachten ist, dass es sich hier wirklich nur um ein Modell handelt, das versucht, die Realität möglichst exakt

wiedergeben. Es handelt sich dabei jedoch nicht um eine physikalisch vollkommen korrekte Beschreibung.

An dieser Stelle werden praktisch nur die von Zybesi verwendeten Formeln mit äußerst knappen Kommentaren, vor allem bei der spiegelnden Beleuchtungskomponente, vorgestellt. Erklärungen und weiterführende Informationen sowie zusätzliche Literatur findet sich in [1].

Die Lichtmenge an einem Punkt der Objektoberfläche hängt im wesentlichen von vier Variablen ab:

- n : Normalenvektor an der Stelle, an der die Lichtintensität bestimmt werden soll, der senkrecht auf dem beleuchteten Objekt steht.
- l : Vektor von dem Punkt, an der die Lichtintensität bestimmt werden soll, in Richtung der Lichtquelle, die hier punktförmig modelliert ist (siehe Abschnitt 2.5).
- a : Vektor von dem Punkt, an der die Lichtintensität bestimmt werden soll, in Richtung des optischen Zentrums der Lochkamera (siehe Abschnitt 2.3).
- h : Halbvektor zwischen a und l , sozusagen die „Winkelhalbierende“ zwischen ihnen.

Alle diese Vektoren werden normiert verwendet, das bedeutet, ihre Länge ist 1 (nach euklidischer Norm): $\|n\| = \|l\| = \|a\| = \|h\| = 1$.

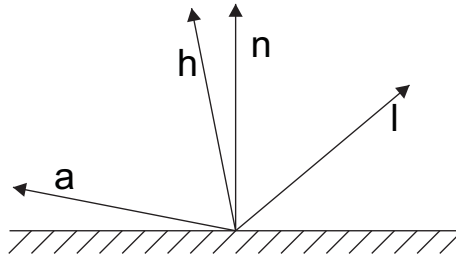


Abbildung 3: a zeigt in Richtung optisches Zentrum der Lochkamera, n steht senkrecht auf der Objektoberfläche, l zeigt in Richtung der punktförmigen Lichtquelle und h ist der Halbvektor zwischen a und l .

Wie man aus diesen vier Vektoren unter zusätzlicher Verwendung von materialspezifischen Konstanten die Beleuchtungsintensität in einem Punkt berechnet, wird nun erläutert.

2.4.1 Ambiente Beleuchtung

Die Lichtmenge an einem Punkt eines beleuchteten Objekts setzt sich prinzipiell aus drei Komponenten zusammen: Diffuse (siehe 2.4.2) und spiegelnde (siehe 2.4.3) Komponente stammen von der Reflexion einer Lichtquelle, während es sich bei der dritten Komponente um Reflexion von Hintergrundlicht handelt. Letztere wird auch

als ambiente oder indirekte Beleuchtung bezeichnet.

Man kann sich diese Beleuchtung als eine Art diffuse Lichtquelle ohne bestimmte Richtung vorstellen. Ihre Ursache ist die Reflexion und Absorption von Licht an der Umgebung. Sie strahlt von allen Seiten homogen auf das zu beleuchtende Objekt und wird von ihm auch homogen in alle Richtungen wieder abgestrahlt. Die Intensität der ambienten Beleuchtung hängt also weder vom Blickwinkel des Betrachters, noch von einer konkreten Lichtquelle ab.

Aus diesem Grund lässt sich die Menge der ambienten Lichtenergie relativ einfach durch folgende Gleichung ausdrücken:

$$I_a = I_v \cdot k_a, I_v \in [0, 1], k_a \in [0, 1]$$

I_a ist dabei die resultierende Menge ambienten Lichts an einem infinitesimal kleinen Flächenstück. k_a bezeichnet den sogenannten ambienten Reflexionskoeffizienten, welcher materialspezifisch ist und ein Maß der Absorption von ambienten Licht durch das Material angibt. I_v steht für die Intensität des ambienten Lichts.

2.4.2 Diffuse Beleuchtung

Die diffuse Reflexion, auch Lambert-Reflexion genannt, tritt bei matten oder stumpfen Oberflächen auf. Sie entsteht durch Unebenheiten der Oberfläche und unterschiedliche Dichten innerhalb des Materials, die das von einer Lichtquelle einfallende Licht innerhalb des Materials hin und her reflektieren, bevor es zu einem Teil wieder an die Umgebung abgegeben wird.

Je senkrechter das Licht der Lichtquelle auf einen Punkt des zu beleuchtenden Objekts trifft, desto größer ist die diffuse Lichtmenge an dieser Stelle, die in alle Richtungen gleich stark abgestrahlt wird. Folgende Gleichung drückt die diffuse Lichtmenge an einem Oberflächenpunkt aus:

$$I_d = I_w \cdot k_d \cdot \langle l | n \rangle, I_w \in [0, 1], k_d \in [0, 1]$$

Analog zur ambienten Beleuchtung ist I_d die resultierende Menge diffusen Lichts an einem infinitesimal kleinen Flächenstück. k_d bezeichnet den diffusen Reflexionskoeffizienten, welcher ebenfalls wie k_a materialspezifisch ist und ein Maß der Absorption von nun diffusem Licht bei den Reflexionen innerhalb des Materials angibt. l ist die Richtung vom betrachteten Oberflächenpunkt zur Lichtquelle, welche mit der Intensität I_w strahlt. n ist der Normalenvektor an dem Punkt der Oberfläche, die gerade betrachtet wird.

2.4.3 Spiegelnde Beleuchtung

Bei der bisher betrachteten diffusen Reflexion wird Licht gleichförmig in alle Richtungen abgestrahlt. Aus diesem Grund ist I_d auch unabhängig von der Richtung des Betrachters a . Für eine realistische Darstellung von Objekten fehlt noch die Modellierung von Glanzlichtern, was nun nachgeholt wird.

Spiegelnde Reflexion entsteht durch die Tatsache, dass eintreffendes Licht von einem Objekt hauptsächlich in eine Richtung abgestrahlt wird. Beim bekannten Sonderfall eines perfekten Spiegels wird Licht nach dem Gesetz „Einfallswinkel gleich Ausfallswinkel“ reflektiert.

Modelliert wird solches Verhalten hier durch sogenannte Microfacetten. Diese Microfacetten kann man sich als winzige, perfekte Spiegel auf der Oberfläche des Materials vorstellen. Weisen alle Facetten dieselbe Richtung wie die gesamte Oberfläche auf, liegt eine glatte bzw. stark glänzende Oberfläche vor. Ist im Gegensatz dazu die Richtung der Microfacetten stark gestreut, wirkt die Oberfläche matt und die Grenze des Glanzlichts verschwimmt.

Die Menge des reflektierten Lichts hängt nun auch von der Position des Betrachters ab und wird über folgende Gleichung modelliert:

$$I_s = I_w \cdot k_s \cdot \frac{D \cdot G \cdot F_\lambda}{\langle l|n \rangle \cdot \langle a|n \rangle}, I_w \in [0, 1], k_s \in [0, 1]$$

Während I_s dabei die resultierende Menge spiegelnden Lichts an einem infinitesimal kleinen Flächenstück ist, bezeichnet k_s den sogenannten spiegelnden Reflexionskoeffizienten, welcher materialspezifisch ist und ein Maß der Absorption von gespiegeltem Licht angibt. I_w ist die Intensität der Lichtquelle.

Eingangs wurde ja bereits das Modell der Microfacetten vorgestellt. Der Parameter $D \in [0, 1]$ beschreibt den Anteil der Microfacetten, die aufgrund ihrer Lage zur Spiegelung beitragen. Dieser Parameter D lässt sich je nach Modell unterschiedlich definieren. Zybesi unterstützt folgende drei Berechnungsvorschriften:

- Methode von Phong:

$$D = \langle l|n \rangle^k, k \geq 0$$

- Methode unter Verwendung der Exponentialfunktion:

$$D = d \cdot e^{-\left(\frac{\phi}{m}\right)^2}$$

- Methode von Beckmann:

$$D = \frac{1}{m^2 \cdot \langle n|h \rangle^4} \cdot e^{-\left(\frac{\tan \phi}{m}\right)^2}$$

Hierbei bezeichnen k und d materialspezifische Konstanten, m das quadratische Mittel der Facettenneigungen.

Die Microfacetten sind auch die Ursache für den Parameter $G \in [0, 1]$, der als geometrischer Abschwächungsfaktor bezeichnet wird. Dieser beschreibt den Grad der gegenseitigen Verschattung der Facetten und der daraus resultierenden Absorption von Licht. Dabei kann Licht bereits beim Einfall gemindert werden (ausgedrückt durch G_l), genau so wie beim Wiederaustritt aus dem Material (ausgedrückt durch G_a).

$$G_l = 2 \cdot \frac{\langle n|h \rangle \cdot \langle n|l \rangle}{\langle l|h \rangle}$$

$$G_a = 2 \cdot \frac{\langle n|h \rangle \cdot \langle n|a \rangle}{\langle a|h \rangle}$$

Aufgrund der Definition des Halbvektors h als Winkelhalbierende zwischen l und a gilt $\langle l|h \rangle = \langle a|h \rangle$. Der geometrische Abschwächungsfaktor G ergibt sich dann aus

$$G = \min \{1, G_l, G_a\}.$$

Zuletzt ist noch der Zweck des Parameters F_λ zu klären. Er verwendet das Reflexionsgesetz nach Fresnel. Grundlage dieses Gesetzes sind neben Einfallswinkel und Beobachtungswinkel der Brechungsindex des Oberflächenmaterials.

Mit $c = \langle l|h \rangle$ und $g = \sqrt{\eta^2 + c^2 - 1}$ (η ist der effektive Brechungsindex des verwendeten Materials, welcher definiert ist als Verhältnis der Lichtgeschwindigkeit im Vakuum zur Lichtgeschwindigkeit im Medium) ergibt sich

$$F_\lambda = \frac{1}{2} \cdot \left(\frac{g-c}{g+c} \right)^2 \cdot \left(1 + \left(\frac{c \cdot (g+c) - 1}{c \cdot (g-c) + 1} \right)^2 \right).$$

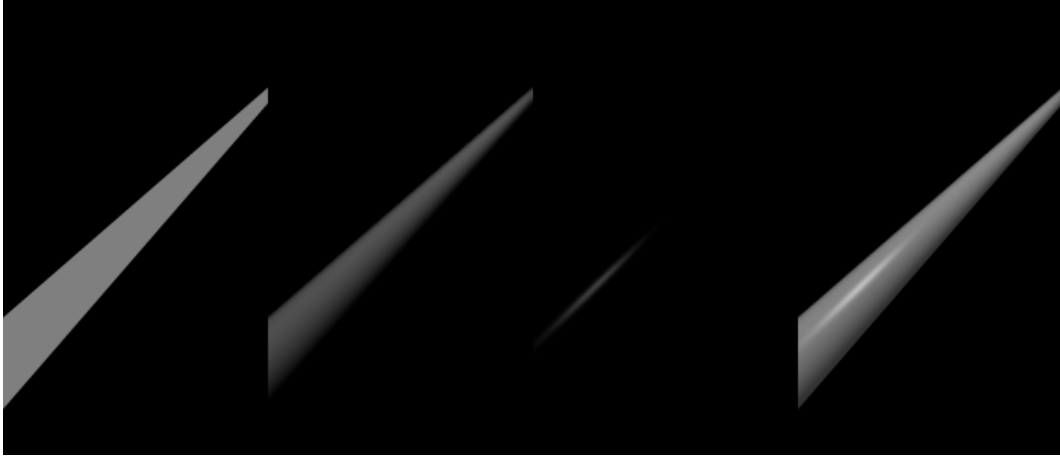


Abbildung 4: Linkes Bild: Ambiente Beleuchtung eines Zylinders im Raum. Zweites Bild von links: Diffuse Beleuchtung des selben Zylinders. Zweites Bild von rechts: Glanzeffekt des Zylinders nach der Methode von Phong. Rechts Bild: Die Summe der drei Beleuchtungskomponenten ergibt das Gesamtbild.

2.4.4 Minderung der Lichtintensität durch Abstand.

Aus der Physik ist bekannt, dass die Energie einer punktförmigen Lichtquelle (wie sie Zybesi verwendet, siehe 2.5) mit dem Kehrwert des Quadrats des Abstandes zwischen Lichtquelle und Oberfläche abnimmt. Dies wird auch als quadratisches Entfernungsgesetz bezeichnet. Es wird folgendermaßen ausgedrückt:

$$C = \min \left\{ 1, \frac{1}{\alpha_0 + \alpha_1 \cdot d + \alpha_2 \cdot d^2} \right\}.$$

C ist also ein Maß für die Minderung von Lichtenergie durch den Abstand. d ist dabei der Abstand des betrachteten Oberflächenpunktes zur punktförmigen Lichtquelle, während α_0 , α_1 und α_2 vom Benutzer festlegbare Konstanten sind. Für den Fall $\alpha_0 = 1$, $\alpha_1 = 0$ und $\alpha_2 = 0$ findet keine Minderung der Lichtenergie durch den Abstand statt.

Diese Abschwächung der Lichtenergie tritt zweimal auf, nämlich einerseits beim Abstand von Lichtquelle zum beleuchteten Objekt (in diesem Fall werden diffuse und spiegelnde Lichtkomponente geschwächt) und andererseits beim Abstand von beleuchtetem Objekt zur Kamera (in diesem Fall werden alle drei Lichtkomponenten geschwächt).

2.4.5 Beleuchtung zusammengefasst

Um nun die Menge der Lichtenergie in einem Oberflächenpunkt zu erhalten, müssen ambiente (2.4.1), diffuse (2.4.2) und spiegelnde (2.4.3) Beleuchtung aufsummiert werden. Für die diffuse und spiegelnde Beleuchtung spielt darüber hinaus auch noch der Abstand von der Lichtquelle (2.4.4) eine Rolle. Somit ergibt sich für die Lichtenergie I in einem Punkt auf der Oberfläche des Objekts:

$$I = I_a + C \cdot (I_d + I_s).$$

Sollen in der Szene mehrere Lichtquellen vorkommen, so muss für jede von ihnen der Wert von I_d und I_s separat berechnet werden. Die Werte pro Lichtquelle werden dann einfach aufsummiert, um die Lichtenergie aller Lichtquellen in einem Oberflächenpunkt zu erhalten.

2.5 Modellierung der Lichtquelle

Wie bereits mehrfach erwähnt, wird die Lichtquelle in Zybesi als punktförmige Lichtquelle modelliert. In der kompletten Szene verwendet Zybesi lediglich eine Lichtquelle.

Eine realistische und damit bessere Modellierung von Lichtquellen wäre ganz bestimmt die Beschreibung der Lichtquelle als Scheibe. Deren Durchmesser könnte man immer noch so weit verringern, dass ihr Verhalten dem einer punktförmigen Lichtquelle entspricht. Jedoch bedeutet diese Maßnahme zusätzlichen Aufwand bei der Implementierung, der bei diesem Programmierpraktikum vermieden wird.

Aus dem selben Grund wird auf die Möglichkeit verzichtet, mehrere Lichtquellen zu definieren. Durch leichte Modifikationen am Quelltext sollte dies jedoch ohne all zu großen Mehraufwand realisierbar sein. Allerdings muss dann beachtet werden, dass die Laufzeit für die Berechnung der diffusen und der spiegelnden Lichtkomponente linear mit der Anzahl der Lichtquellen ansteigt.

2.6 Schnitt von Sichtstrahl mit dem Zylinder

2.4 hat gezeigt, wie man die Beleuchtungsintensität an einem Punkt der Oberfläche des beleuchteten Objekts berechnet. Man möchte jedoch wissen, wie stark die Beleuchtungsintensität an einem Punkt im Sensorarray ist, da diese Information für die Bestimmung des Grauwertes im zu erstellenden Bild benötigt wird. Zu diesem Zweck verfolgt man einen Sichtstrahl, der durch den Punkt b in der Retinaebene und dem optischen Zentrum c der Lochkamera definiert ist, bis dieser den Zylinder im Punkt x schneidet und berechnet an dieser Stelle, wie besprochen, die Beleuchtungsintensität. b und c legen die Richtung s des Sichtstrahls fest. Die Beleuchtungsintensität im Punkt b in der Retinaebene entspricht der Beleuchtungsintensität auf der Zylinderoberfläche im Punkt x geschwächt durch den Abstand zwischen b und x (siehe 2.4.4). Die Bestimmung des Schnittpunktes x von Sichtstrahl und Zylinder wird nun vorgestellt.

Um den Schnittpunkt x zu erhalten, stellt man zunächst die Gleichung des Zylinders und die Geradengleichung des Sichtstrahls auf.

$$(1) \ x = d + r \cdot \cos(\varphi) \cdot i + r \cdot \sin(\varphi) \cdot j + \lambda \cdot f$$

$$(2) \ x = b + \mu \cdot s$$

r ist dabei, wie gehabt der Radius des Zylinders, f ist die Richtung der Hauptachse des Zylinders und d ist der Durchstoßpunkt der Hauptachse durch die xy -Ebene. $i, j \in \mathbb{R}^3$ mit $\|i\| = \|j\| = 1$ sind zwei untereinander und auf f orthogonale Vektoren mit Länge 1. Also gilt $\langle i|j \rangle = \langle i|f \rangle = \langle j|f \rangle = 0$.

Da i und j nicht eindeutig bestimmt sind, lässt sich φ passend wählen. Wir setzen $\varphi = 0$.

$$(1') \ x = d + r \cdot i + \lambda \cdot f$$

Als nächsten Schritt setzen wir (1') und (2) gleich und bringen den Vektor i auf die linke Seite.

$$r \cdot i = b - d + \mu \cdot s + \lambda \cdot f$$

Wir haben nun eine Gleichung mit zwei Veränderlichen (μ und λ) und dem Vektor i . Um diesen zu eliminieren wird nun ein kleiner Trick angewandt: Einmal eliminieren wir i durch Multiplikation mit f ($\langle i|f \rangle = 0$) und einmal durch Multiplikation mit sich selbst ($\langle i|i \rangle = 1$). Somit erhält man zwei skalare Gleichungen in μ und λ .

$$(3) \ 0 = \langle b|f \rangle - \langle d|f \rangle + \mu \cdot \langle s|f \rangle + \lambda \cdot \langle f|f \rangle$$

$$(4) \ r^2 = (b - d + \mu \cdot s + \lambda \cdot f)^2$$

Löst man (3) nach λ auf und setzt sie in (4) ein, so erhält man

$$r^2 = ((b - d) + (\mu \cdot s - \frac{\langle f|b-d \rangle + \mu \cdot \langle s|f \rangle}{\langle f|f \rangle} \cdot f))^2.$$

Für den nächsten Schritt werden zur Vereinfachung drei Substitutionen benötigt.

$$sub_1 = \frac{\langle f|b-d \rangle}{\langle f|f \rangle} \cdot f$$

$$sub_2 = \frac{\langle s|f \rangle}{\langle f|f \rangle} \cdot f$$

$$sub_3 = b - d$$

Nach mehrmaligem Anwenden der Binomischen Formel, Ausklammern und Zusammenfassen von konstantem, linearem und quadratischem Term erhält man folgende quadratische Gleichung, die nun recht einfach nach μ aufzulösen ist.

$$\begin{aligned} & \mu^2 \cdot (\langle s|s \rangle - 2 \cdot \langle s|sub_2 \rangle + \langle sub_2|sub_2 \rangle) + \\ & + 2\mu \cdot (\langle s|sub_1 \rangle - 2 \cdot \langle sub_1|sub_3 \rangle - 2 \cdot \langle s|sub_2 \rangle + 2 \cdot \langle sub_2|sub_3 \rangle) + \\ & + (\langle sub_1|sub_1 \rangle - 2 \cdot \langle sub_1|sub_2 \rangle + \langle sub_2|sub_2 \rangle - r^2) = 0 \end{aligned}$$

Das Lösen der quadratischen Gleichung liefert in der Regel zwei Ergebnisse für μ . Man muss dasjenige μ wählen, das zu dem Schnittpunkt x gehört, der von der Kamera aus sichtbar ist und nicht dasjenige μ , dass zu dem Schnittpunkt gehört, der „hinten“ am Zylinder liegt.

Setzt man μ nun in (1) oder besser (weil einfacher) in (2) ein, erhält man den Schnittpunkt x von Sichtstrahl und Zylinder.

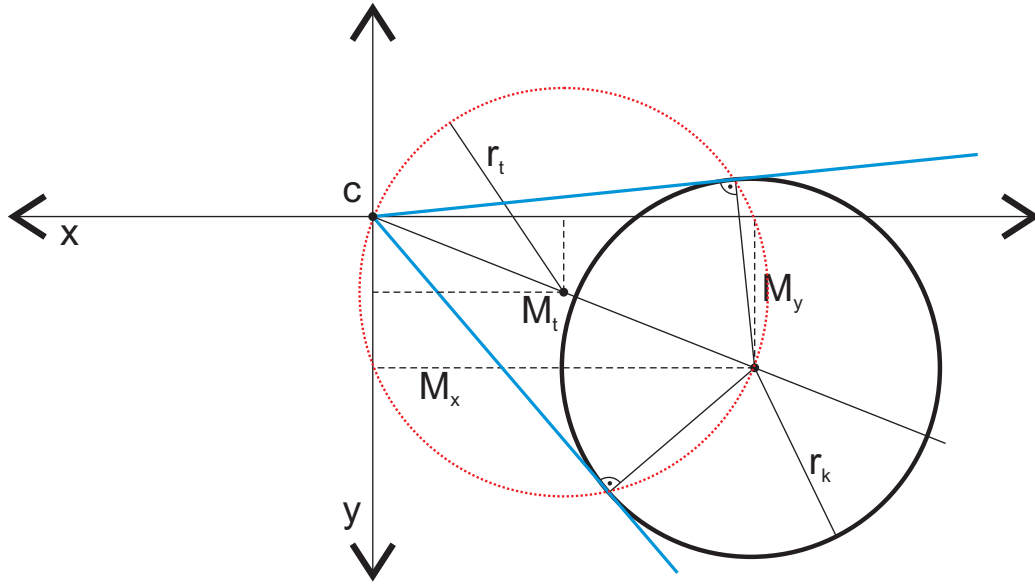


Abbildung 5: Skizze der Tangentialpunktbestimmung in der xy -Ebene. Der rote Kreis ist der erwähnte Thaleskreis, die blauen Geraden die Tangenten an den Schnittkreis des rotierten Ausgangszyinders mit der xy -Ebene, der schwarz eingezeichnet ist.

2.7 Bestimmung der beiden Schattengrenzlinien des Zylinders

Aus Gründen der Performancesteigerung ist es nützlich zu wissen, wo die Projektion der Schattengrenzlinien des Zylinders in der Retinaebene verlaufen. Unter Schattengrenzlinien des Zylinders versteht man die beiden Mantellinien, die die Grenze zwischen Vorder- und Hinterseite des Zylinders bilden. Sie sind gerade noch sichtbar und könnten auch als „Rand“ des Zylinders bezeichnet werden, da ihre Projektion in die Retinaebene die Grenze zwischen Zylinderprojektion und Hintergrundprojektion bildet. Ein Zylinder besitzt zwei von diesen Schattengrenzlinien.

Um die beiden Schattengrenzlinien zu erhalten reicht es aus, zwei beliebige Punkte $tp_1, tp_2 \in \mathbb{R}^3$ auf diesen Schattengrenzlinien im Raum zu bestimmen. Diese verwendet man dann als Aufpunkte der Schattengrenzlinien. Die Richtung der Schattengrenzlinien entspricht dabei der Richtung der Zylinderhauptachse f . tp_1 und tp_2 sind also zwei Berührungspunkte an den Zylinder die auf Sichtstrahlen liegen, die den Zylinder berühren.

Eine Möglichkeit der Ermittlung der beiden Tangentialpunkte ist, den Zylinder so zu drehen, dass seine Zylinderhauptachse senkrecht zur xy -Ebene steht ($f \perp u_1$ und $f \perp u_2$). Der Zylinder schneidet dann die xy -Ebene in einem Kreis, dessen Radius der Radius des Zylinders r ist und dessen Mittelpunkt als $M_k = (M_x, M_y)$ bezeichnet wird. Im Allgemeinen entspricht dieser Kreismittelpunkt M_k nicht dem Durchstoßpunkt des Ausgangszyinders durch die xy -Ebene d . Somit hat man die Bestimmung der Tangentialpunkte im Raum auf das Problem der Bestimmung der Tangentialpunkte an den Kreis $(x - M_x)^2 + (y - M_y)^2 = r^2$ in der xy -Ebene reduziert.

An dieser Stelle bedient man sich des Hilfsmittels des Thaleskreises. Die Mitte zwi-

schen Koordinatenursprung c und M_k wird der Mittelpunkt des Thaleskreises M_t . Als Radius des Thaleskreises r_t wählt man die Länge der Strecke zwischen M_t und c , die gleich der Länge der Strecke zwischen M_t und M_r ist. Alle Punkte, die auf diesem Thaleskreis liegen, bilden zusammen mit den beiden Punkten c und M_k ein rechtwinkliges Dreieck. Somit ergibt der Schnitt des Thaleskreises mit dem Kreis $(x - M_x)^2 + (y - M_y)^2 = r^2$ zwei Tangentialpunkte in der xy -Ebene. Diese beiden Punkte werden nun zurück in den Raum gedreht und zwar auf umgekehrte Weise, wie zuvor der Ausgangszylinder gedreht wurde, sodass seine Hauptachse senkrecht zur xy -Ebene steht. tp_1 und tp_2 entsprechen nun diesen beiden in den Raum zurückgedrehten Punkten.

2.8 Das Sensormaaß

Eine Digitalkamera wertet nicht uniform auf der ganzen Sensorapperturfläche aus. In der Regel trägt Licht, das im Zentrum des Sensors auftrifft, einen höheren Beitrag zur Lichtleistungsdichte über dem Sensor bei, als Licht, das am Rand des Sensors auftrifft. Darüber hinaus können Sensoren, die neben dem gerade betrachteten liegen, ebenfalls einen Anteil zur Gesamtlichtleistungsdichte beitragen. Dieser Umstand muss bei einer realistischen Modellierung beachtet werden und wird durch das Sensormaaß ausgedrückt.

Beim Sensormaaß handelt es sich also um eine Funktion über der Fläche des Pixelsensors die angibt, wie groß der Anteil des an einer bestimmten Stelle auftreffenden Lichts an der Gesamtlichtleistungsdichte ist. In der Regel haben derartige Funktionen eine Glockenform mit der Spitze über dem Zentrum des Sensors, da dort der Sensor am empfindlichsten ist. Diese Empfindlichkeit nimmt in den Randbereichen des Sensors ab.

In der Praxis hat es sich bewährt, das Sensormaaß mit B-Spline-Flächen zu approximieren. An dieser Stelle sei lediglich die Definition von B-Splines erwähnt. Mehr über B-Splines kann [3] entnommen werden.

Die B-Spline Basisfunktion $N_{i,p,\tau}(t)$ der Ordnung p wird durch die Rekursionsformel nach Boor/Cox/Mansfield definiert. Für sie wird ein Knotenvektor $\tau = (\tau_1, \dots, \tau_n)$, $n \geq 2p$, auch Knickstellensequenz genannt, benötigt. Die Rekursionsformel lautet wie folgt:

$$N_{i,1,\tau}(t) = \begin{cases} 1 & , u \in [\tau_i, \tau_{i+1}] \\ 0 & , \text{sonst} \end{cases}$$

und

$$N_{i,p,\tau}(t) = \frac{t - \tau_i}{\tau_{i+p-1} - \tau_i} \cdot N_{i,p-1,\tau}(t) + \frac{\tau_{i+p} - t}{\tau_{i+p} - \tau_{i+1}} \cdot N_{i+1,p-1,\tau}(t).$$

Daraus lassen sich nun B-Spline-Kurven und B-Spline-Flächen definieren.

Eine B-Spline-Kurve $C(t) : [\tau_p, \tau_{n-p+1}] \mapsto \mathbb{R}_0^+$ der Ordnung p mit Knickstellensequenz τ und Kontrollpunkten P_i , $i \in \mathbb{Z}_{1,n-p}$ (auch DeBoor-Punkte oder B-Spline-Koeffizienten genannt) wird definiert durch

$$C(t) = \sum_{i=1}^{n-p} P_i \cdot N_{i,p,\tau}(t).$$

Analog dazu wird eine B-Spline-Fläche $C(s, t) : [\tau_p, \tau_{n-p+1}] \times [\mu_q, \mu_{m-q+1}] \mapsto \mathbb{R}_0^+$ der Ordnung p und q mit Knickstellensequenz $\tau = (\tau_1, \dots, \tau_n)$, $n \geq 2p$ und $\mu = (\mu_1, \dots, \mu_m)$, $m \geq 2q$ und Kontrollpunkten P_{ij} , $(i, j) \in \mathbb{Z}_{1,n-p} \times \mathbb{Z}_{1,m-q}$ definiert als

$$C(s, t) = \sum_{i=1}^{n-p} \sum_{j=1}^{m-q} P_{ij} \cdot N_{i,p,\tau}(s) \cdot N_{j,q,\mu}(t).$$

In Zybesi wurde sowohl für die x - als auch die y -Richtung die selbe Knickstellensequenz verwendet. Somit ist $\tau = \mu$ und die B-Spline-Fläche ist über dem Quadrat $[\tau_p, \tau_{n-p+1}]^2$ definiert.

2.9 Numerische Integration der Lichtleistungsdichte

Bisher war immer die Rede davon, einen Sichtstrahl, der von der Lochkamera ausgeht, mit dem Zylinder zu schneiden (siehe 2.6) und an dieser Schnittstelle die Beleuchtungsintensität auszurechnen (siehe 2.4). Um jedoch realistisches Verhalten zu simulieren muss beachtet werden, dass die Digitalkamera zur Bestimmung des Grauwertes eines Bildpunktes die Lichtenergie misst, die auf der gesamten Fläche des zum Bildpunkt korrespondierenden Sensors auftrifft. Es findet also eine Integration über der gesamten Sensorfläche statt.

Allerdings führt Zybesi eine derartige Integration nicht durch, sondern greift stattdessen auf eine numerische Integration zurück. Dies hat den Vorteil, dass man nicht die exakten Integrale der ambienten (I_a), diffusen (I_d) und spiegelnden (I_s) Lichtleistungsfunktion bestimmen muss. Jedoch handelt es sich bei einer numerischen Integration immer nur um eine Approximation an das tatsächliche Integral.

Als numerische Integration verwendet Zybesi die Simpson-Regel [2]. Bei der Simpson-Regel handelt es sich um eine Quadraturformel die eine Menge von äquidistanten Stützstellen der Funktion benötigt, deren Integral sie approximieren soll. Dabei kann das Integral nur über einem beschränkten Intervall berechnet werden. Anwenden lässt sich die Simpson-Regel auf Funktionen mit einer beliebigen Anzahl von Veränderlichen, also der Form $f : \mathbb{R}^n \mapsto \mathbb{R}, n \in \mathbb{N}_1$. An dieser Stelle sei kurz die Simpson-Regel für Funktionen mit einer und, wie sie in Zybesi benötigt wird, zwei Veränderlichen vorgestellt.

Sei $a \in \mathbb{R}$ die untere und $b \in \mathbb{R}$ die obere Integrationsgrenze (es wird also über dem Intervall $I = [a, b]$ integriert) sowie $m \in \mathbb{N}_1$ die Anzahl der $h = \frac{b-a}{m}$ großen Intervalle (daraus folgt, dass $m+1$ Stützstellen benötigt werden), so ist das Simpsonfunktional definiert als

$$s_m^1(f) = \frac{h}{3} \cdot (f(a) + 4 \cdot f(a+h) + 2 \cdot f(a+2h) + 4 \cdot f(a+3h) + \dots + 4 \cdot f(b-h) + f(b)).$$

Dabei wird davon ausgegangen, dass $f : \mathbb{R} \mapsto \mathbb{R}$ eine Funktion in einer Veränderlichen ist, die integriert werden soll. Für die numerische Integration reicht es also aus, die Funktionswerte an den Stellen $a + i \cdot h$, $i \in \mathbb{Z}_{0,m}$ zu kennen. Anhand der Definition des Simpsonfunktionals lässt sich erkennen, dass die untere Integrationsgrenze a die erste und die obere Integrationsgrenze b die letzte Stützstelle ist.

Aus $s_m^1(f)$ lässt sich leicht das Simpsonfunktional für Funktionen in zwei Veränderlichen $s_m^2(f)$ bilden. Dabei sind $a, b \in \mathbb{R}$ mit $a < b$ die Integrationsgrenzen in x -Richtung, $c, d \in \mathbb{R}$ mit $c < d$ die Integrationsgrenzen in y -Richtung. Es wird also über dem

Rechteck $[a, b] \times [c, d]$ integriert. Die Anzahl der Intervalle in y -Richtung ist n . Jedes dieser Intervalle ist $g = \frac{d-c}{n}$ groß.

$$s_{m,n}^2(f) = \frac{h}{3} \cdot (s_m^1(f_0) + 4 \cdot s_m^1(f_1) + 2 \cdot s_m^1(f_2) + 4 \cdot s_m^1(f_3) + \dots + 4 \cdot s_m^1(f_n) + s_m^1(f_{n+1}))$$

wobei $f_j = f(x, c + j \cdot g)$ für $j \in \mathbb{Z}_{0,n}$ und $x \in [a, b]$. $f : \mathbb{R}^2 \mapsto \mathbb{R}$ ist nun also eine Funktion in zwei Veränderlichen.

Da bei Zybesi davon ausgegangen wird, dass die Sensorflächen quadratisch sind, können, wie schon in 2.8, einige Vereinfachungen gemacht werden: $a = c$, $b = d$, $m = n$ und daraus folgend $g = h$. Somit integriert Zybesi immer über der Fläche $[a, b]^2$.

3 Vorgehen in der Praxis

Zybesi unterstützt zwei Modi zur Erstellung von Simulationsbildern eines beleuchteten Zylinders: Einerseits können Bilder via Raytracing erstellt werden, wie man dies auch aus Programmen wie Pov-Ray [4] kennt. Andererseits ist es möglich, eine realistischere Variante zu wählen, die die in 2.8 und 2.9 vorgestellten Verfahren beinhaltet.

In diesem Kapitel werden die Implementierungen der beiden Varianten „Raytracing“ und „Realistische Modellierung“ vorgestellt.

3.1 Raytracing

Beim Raytracing wird jeder Bildpunkt des zu erstellenden Bildes sequenteill auf dieselbe Weise abgearbeitet.

1. Zunächst wird das Pixelzentrum b des zum aktuell bearbeiteten Bildpunkt korrespondierenden Sensors der virtuellen Lochkamera ermittelt.
2. Von dort aus „schießt“ man einen Sichtstrahl mit Richtung s durch das optische Zentrum c der Lochkamera und bringt diesen Sichtstrahl nach dem in 2.6 vorgestellten Verfahren mit dem Zylinder zum Schnitt. Dieser Schnittpunkt heißt x .
3. In diesem Punkt berechnet man dann nach dem in 2.4 eingeführten Beleuchtungsmodell die Lichtintensität. Diese Lichtintensität wird dann dafür verwendet, um den Grauwert des gerade bearbeiteten Bildpunktes zu ermitteln. Zu beachten ist noch, dass eine Minderung der Lichtenergie durch den Abstand von Zylinder zur Lichtquelle und Zylinder zur Kamera entsteht.

3.2 Realistische Modellierung

Die Erstellung eines realitätsnaheren Bildes ähnelt in einigen Punkten dem Vorgehen beim Raytracing, jedoch werden zusätzliche Aspekte beachtet.

1. Im ersten Schritt werden die Schattengrenzlinien nach dem in 2.7 vorgestellten Verfahren ermittelt und in die Retinaebene projiziert. Diese Information kann benutzt werden, damit nur diejenigen Stützstellen für die numerische Integration berechnet werden, die auch tatsächlich von der Projektion des Zylinders in die Retinaebene betroffen sind.
2. Nachdem man nun weiß, welche Stützstellen von der Projektion des Zylinders betroffen sind, kann man für diese Stützstellen via Raytracing (wie in 3.1) die Lichtleistungsdichte an den entsprechenden Punkten berechnen. Alle Stützstellen, die innerhalb der beiden Schattengrenzlinien in der Retinaebene liegen, werden in einem großen Array gespeichert.
3. Nun kann das zu erstellende Bild wieder Bildpunkt für Bildpunkt durchlaufen werden, um für den aktuellen Pixel den korrekten Grauwert zu berechnen. Es wurde ja bereits erwähnt, dass man, um eine realistische Modellierung einer Kamera zu erreichen, beachten muss, dass nicht ausschließlich die Lichtmenge über dem Sensor selbst für den Grauwert des korrespondierenden Bildpunktes verantwortlich ist, sondern auch benachbarte Sensoren diesen Grauwert beeinflussen können. Deshalb werden auch die nächsten und evtl. übernächsten Nachbarn des gerade bearbeiteten Bildpunktes beachtet. Die Stützstellen für die numerische Integration werden mit dem Sensormaß (siehe 2.8) versehen, da der Sensor in seiner Mitte stärker auswertet als am Rand und der Anteil der Energie, die von benachbarten Sensoren auf den gerade betrachteten einwirkt, sehr gering ist. Anschließend wird auf die Stützstellen die Simpson-Regel angewandt, die den Grauwert für den gerade betrachteten Bildpunkt liefert.

4 Besonderheiten und Erkenntnisse

Ziel dieses Programmierpraktikums ist es auch, Erfahrungen mit den verwendeten Modellen, vor allem mit dem Beleuchtungsmodell von Blinn, zu sammeln. Einige Erkenntnisse seien nun an dieser Stelle präsentiert.

4.1 Überlauf bei der Berechnung des spekularen Anteils

Bei der Vorstellung des Beleuchtungsmodells (siehe 2.4) wurde festgelegt, dass keiner der drei Werte I_a , I_d und I_s einen Wert größer 1 annimmt. Für den ambienten und den diffusen Beleuchtungsanteil leuchtet dies mit einem bloßen Blick auf die entsprechende Berechnungsvorschrift schnell ein. Für den spiegelnden Beleuchtungsanteil lässt sich dies jedoch nicht so einfach zeigen.

Betrachtet man die von Zybesi berechneten Werte für I_s (Berechnungsvorschrift siehe 2.4.3) fällt schnell auf, dass durchaus Werte für I_s zustande kommen, die größer sind als 1. Dieses Phänomen tritt sowohl bei der phong'schen Berechnungsmethode für den Faktor D auf, als auch bei der beckmann'schen oder der exponentiellen. Da selbst Werte für I_s auftreten, die größer als $\frac{11}{10}$ sind, kann man auch davon ausgehen, dass weder Rundungsfehler noch numerische Instabilitäten Ursache dieses Phänomens

sind. Diese beiden Fehlerquellen kann man wohl auch deshalb ausschließen, da dieser Fehler sehr häufig vorkommt, also nicht nur in Randbereichen, beispielsweise wenn Vektoren beinahe die selbe oder entgegengesetzte Richtung haben. Die tatsächliche Ursache liegt wohl viel mehr in einem Missverhältnis von Nenner und Zähler bei der Berechnung von I_s . Während der geometrische Abschwächungsfaktor G noch ausschließlich mit Skalarprodukten der verwendeten Vektoren a , l , n und h arbeitet, greifen die Faktoren D und F_λ auf die Tangens- und Exponentialfunktion zurück oder bringen weitere Parameter, wie den Brechungsindex ins Spiel. Besondere Erwähnung bedarf es noch des Faktors D , wenn er nach der beckmann'schen Methode berechnet wird. In diesem Fall nimmt nämlich selbst der Faktor D keine Werte mehr an, die im Intervall $[0, 1]$ liegen, sondern teilweise deutlich größere Werte, wie $\frac{14}{10}$, die wiederum keine Seltenheit sind. Eine Möglichkeit, dieses Phänomen in den Griff zu bekommen wäre die Bestimmung des Maximums von I_s bzw. D , wenn es nach der beckmann'schen Methode berechnet wird und die Normierung von I_s und D mit diesem Maximum. Zybesi wendet eine wesentlich einfachere Methode an: Er bildet schlichtweg das Minimum $\min\{I_s, 1\}$ bzw. $\min\{D, 1\}$ und verwendet es für die folgenden Berechnungen. Diese Minimumsbildung kann jedoch explizit über die grafische Benutzeroberfläche durch Aktivieren der Option „Überstrahlung zulassen“ deaktiviert werden. Ist dies der Fall, verwendet Zybesi sowohl für I_s als auch für D nach der beckmann'schen Methode Werte größer als 1, falls derartige Werte auftreten. Abschließend muss man jedoch sagen, dass dieses Überlaufphänomen zwar unabhängig von der Richtung der Vektoren a , l , n und h auftritt, keinesfalls aber von den materialspezifischen Parametern k (für D nach der phong'schen Methode), d (für D nach der exponentiellen Methode), m (für D nach der exponentiellen und der beckmann'schen Methode) und η (für das Reflexionsgesetz nach Fresnel). Wählt man diese realitätsnah, also entsprechend der Parameter von beispielsweise metallischen Oberflächen, tritt das Überlaufphänomen nicht mehr auf. Allerdings ist es beunruhigend zu wissen, dass die Formeln des Beleuchtungsmodells mathematisch nicht einwandfrei sind.

4.2 Harte Kanten an Beleuchtungsgrenzen

Die spiegelnde Beleuchtungskomponente ist für ein weiteres Phänomen verantwortlich. Bei äußerst matten Oberflächen, die die Glanzlichtkomponente weit streuen, kommt es zu dem Effekt, dass an den Beleuchtungsgrenzen der Oberfläche die spiegelnde Lichtkomponente schlagartig abfällt und so eine Kante im Simulationsbild entsteht.

Unter Beleuchtungsgrenze versteht man dabei den Teil des Zylinders, der die Grenze zwischen dem Bereich, der von der Lichtquelle gerade noch beschienen wird und dem Bereich, der bereits auf der „Schattenseite“ des Zylinders liegt, bildet. Begründen lässt sich diese Erscheinung damit, dass die spiegelnde Komponente nur für den Fall $\langle n|l \rangle \in [0, 1]$ berechnet wird. Dies ist auch vollkommen logisch, da ein Glanzlicht nicht an einer Stelle auftreten kann, die nicht von der Lichtquelle beleuchtet wird. Ist die Oberfläche sehr matt, so ist die Glanzlichtkomponente relativ hoch, je weiter man in den Bereich der Beleuchtungsgrenze kommt. Hat man diese überschritten, wird der Zylinder an dieser Stelle nicht mehr beleuchtet, und die spiegelnde Kom-

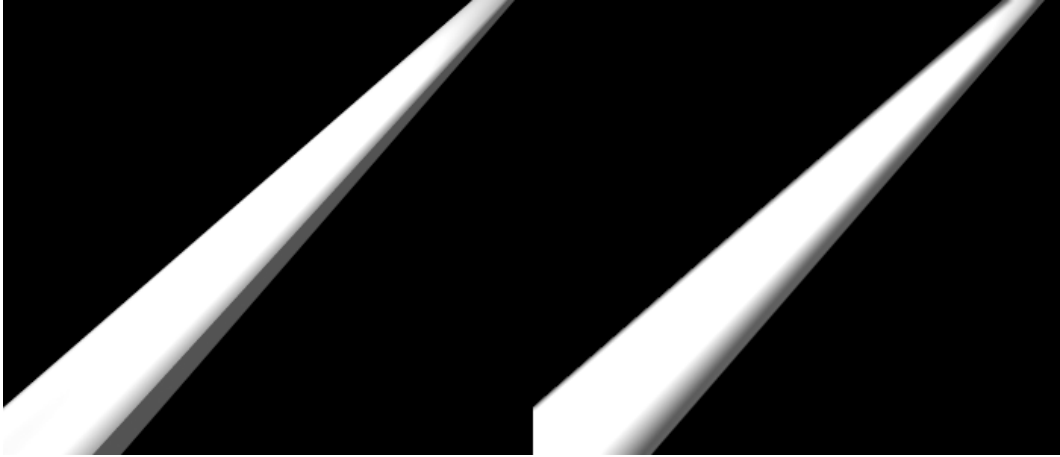


Abbildung 6: Im linken Bild sieht man die harte Kante zwischen dem Glanzlicht (weiß) und dem ambient beleuchteten Zylinderanteil (grau) sehr deutlich. Zur Demonstration dieses Effekts wurde eine äußerst matt spiegelnde Oberfläche verwendet (phong'sche Methode mit $k = 1$) und gleichzeitig die Glanzlichtintensität extrem hoch eingestellt ($I_w = k_s = 1$). Das rechte Bild enthält keine solche Kante, da die spiegelnde Komponente auch für Oberflächenpunkte bestimmt wurde, die gar nicht von der Lichtquelle aus sichtbar sind. Derartige Bilder lassen sich mit Zybesi nicht erstellen.

ponente fällt auf 0 ab.

Interessant ist jedoch, dass wenn man den spiegelnden Anteil auch für Punkte berechnet, die nicht von der Lichtquelle ausgeleuchtet werden, sich dann ein weicher Übergang der spiegelnden Beleuchtungskomponente ergibt. Dies unterstützt Zybesi jedoch nicht, da dieser Fall physikalisch nicht korrekt ist. Nicht beleuchtete Oberflächenpunkte können auch kein Licht reflektieren.

4.3 Laufzeit der Bildgenerierung

Ein gravierendes Problem von Zybesi in der aktuellen Version ist dessen Geschwindigkeit bei der Berechnung der Simulationsbilder. Je nach Ausstattung des Endsystems dauert die Erstellung eines Bildes in industrieüblicher Auflösung (1280×1024 Bildpunkte) im realistischen Bildberechnungsmodus bis zu einer Minute.

Dieser Wert hängt jedoch stark von der Größe der Projektion des Zylinders in der Retinaebene und dem Anteil dieser Projektion im Simulationsbild ab. Für Bildpunkte, die im Simulationsbild schwarzen Hintergrund zeigen, werden logischerweise keine Beleuchtungsberechnungen durchgeführt. Aus diesem Grund ist es auch hilfreich zu wissen, wo die Schattengrenzlinien des Zylinders in der Retinaebene verlaufen. Auf diese Weise erspart man sich zusätzlich die Bestimmung des Schnittpunktes von Sichtstrahl und Zylinder für Sichtstrahlen, die den Zylinder gar nicht schneiden. Man sieht also, je kleiner die Zylinderprojektion im Simulationsbild, desto weniger Berechnungen sind notwendig.

Ein weiterer Faktor, der der Performancesteigerung dient, ist das Vermeiden der Berechnung des ambienten Lichtanteils falls $I_v = 0 \vee k_a = 0$, des diffusen Lichtanteils

falls $I_w = 0 \vee k_d = 0$ und des spiegelnden Lichtanteils falls $I_w = 0 \vee k_s = 0$. Jedoch lässt sich durch diese Maßnahme in der Praxis wohl keine Laufzeit einsparen, da zur Erstellung von realistischen Simulationsbildern alle drei Beleuchtungskomponenten benötigt werden.

Hauptursache für die lange Laufzeit ist die Bestimmung der Stützstellen, die für die numerische Integration benötigt werden. Je nachdem wie der Anwender Zybesi konfiguriert hat (drei, fünf, sieben usw. Stützstellen pro Dimension und Bildpunkt) müssen pro Sensor $3^2 = 9$, $5^2 = 25$, $7^2 = 49$ usw. Stützstellen ausgerechnet werden. Eine geringe Anzahl von Stützstellen bedeutet zwar eine bessere Laufzeit, jedoch einen größeren Fehler bei der Approximation des tatsächlichen Lichtleistungsdichteintegrals durch die numerische Integration. Eine höhere Anzahl von Stützstellen verringert diesen Approximationsfehler, was man sich jedoch mit deutlich längerer Laufzeit erkauft.

Abschließend muss jedoch gesagt werden, dass die Laufzeitoptimierung von Zybesi nicht im Mittelpunkt dieses Programmierpraktikums steht und deshalb noch großes Optimierungspotential vorhanden sein dürfte.

5 Die Software Zybesi

Nachdem im vorherigen Kapitel erläutert wurde, wie Zybesi unter Verwendung der Modelle von Zylinder, Beleuchtung und Lochkamera Simulationsbilder erstellt, ist es nun an der Zeit, einen genaueren Blick auf die Software selbst zu werfen. Dazu gehört sowohl der Quelltext als auch das fertig kompilierte Programm.

5.1 Anmerkungen zum Quelltext

Zybesi ist in der Programmiersprache C geschrieben. Der Quelltext besteht aus sieben Quelltextdateien, die nun genauer vorgestellt werden sollen. Hilfreich ist auch ein Blick in den Quelltext selbst, da dieser ausführlich kommentiert ist.

- **main.c**: Wie der Name schon sagt, handelt es sich um die Hauptquelltextdatei von Zybesi. Sie enthält neben der `main()`-Funktion auch Anweisungen zur Erstellung der GUI-Elemente und vereint die Funktionalität der anderen sechs Quelltextdateien, um wie in Kapitel 3 beschrieben Simulationsbilder zu erstellen.
Lines of Code: 1553.
Anzahl der Funktionen: 41.
- **bspline.c**: Diese Quelltextdatei stellt Funktionen rund um B-Splines zur Verfügung. Dabei handelt es sich um Funktionen für B-Spline-Kurven und -Flächen und deren Auswertung an bestimmten Stellen, um das Sensormaß zu approximieren.
Lines of Code: 243.
Anzahl der Funktionen: 4.
- **cylinder.c**: Aufgaben, wie die Bestimmung der Richtung der Zylinderhauptachse oder die Berechnung des Schnittpunktes eines Sichtstrahls mit dem Zylinder, bewältigen die Funktionen dieser Quelltextdatei.

Lines of Code: 620.

Anzahl der Funktionen: 6.

- **helper.c**: Die Funktionalität der Funktionen dieser Quelltextdatei sind so allgemein, dass sie auch in Projekten unabhängig von Zybesi Verwendung finden können. Dazu gehören Operationen, wie das Rotieren von Punkten um Koordinatenachsen oder um beliebige Geraden genau so, wie das Lösen von quadratischen Gleichungen oder die orthogonale Projektion von Punkten auf Linien.

Lines of Code: 986.

Anzahl der Funktionen: 31.

- **integration.c**: Für die Bestimmung des Grauwertes eines Bildpunktes bei realistischer Berechnung kommt numerische Integration in Form der Simpson-Regel zur Anwendung. Die dafür benötigte Implementierung findet sich in dieser Quelltextdatei.

Lines of Code: 132.

Anzahl der Funktionen: 2.

- **radiance.c**: Alle für das Beleuchtungsmodell nach Blinn benötigten Funktionen werden von dieser Quelltextdatei bereit gestellt. Dabei handelt es sich hauptsächlich um die Berechnung des ambienten, diffusen und spiegelnden Beleuchtungsanteils in einem Punkt.

Lines of Code: 449.

Anzahl der Funktionen: 10.

- **sensorarray.c**: Besondere Aufmerksamkeit bedarf die Modellierung der Lochkamera. Neben der Lochkameraabbildung müssen Aufgaben wie das Testen, ob eine Linie innerhalb des lichtempfindlichen Bereichs der virtuellen Lochkamera liegt, bewerkstelligt werden. Diese Aufgaben werden von den Funktionen dieser Quelltextdatei ausgeführt.

Lines of Code: 194.

Anzahl der Funktionen: 4.

Mit Ausnahme von **main.c** besitzen alle Quelltextdateien eine Headerdatei, in der die Signaturen der einzelnen Funktionen festgelegt sind.

5.2 Kompilierung des Quelltextes und Kompatibilität

Zybesi macht massiven Gebrauch der nicht freien FORWISS-Libraries [5]. Diese werden einerseits zum Erstellen der GUI verwendet (dabei bauen die FORWISS-Libraries auf GTK [6] auf) und andererseits zum Rechnen mit Punkten und Vektoren in der Ebene und im Raum. Aus diesem Grund müssen die FORWISS-Libraries und GTK sowohl zur Compile- als auch zur Laufzeit auf dem Zielsystem vorhanden sein.

Sowohl unter Linux (32- und 64-Bit Version) als auch unter Windows (32-Bit Version) auf x86-Systemen wurde Zybesi mit GCC [7] bzw. MinGW [8] kompiliert. Für diese beiden Betriebssysteme kann also garantiert werden, dass die Software ohne Fehler läuft, insofern die FORWISS-Libraries und GTK vorhanden sind. Jedoch

dürfte Zybesi auf jedem Mainstreambetriebssystem lauffähig sein, für das GCC, GTK und die FORWISS-Libraries zur Verfügung stehen. Zur Kompilierung unter Linux in den CIP-Pools der Universität Passau liegt dem Quelltext ein Makefile bei. Unter Linux gibt es gegenüber der Windowsversion von Zybesi eine kleine Einschränkung. Das Speichern der Simulationsbilder als PNG-Datei ist nur unter Windows möglich. Aus diesem Grund muss auf Windows-System zusätzlich zu GTK und des FORWISS-Libraries libpng [9] vorhanden sein. Die GCC-Compileranweisung zum Kompilieren lautet, falls man sich im selben Verzeichnis wie die Quelltextdateien befindet:

```
1: gcc -I<Pfad zu den FORWISS-Headerdateien> -O3 -g3 -Wall -c
-fmessage-length=0 -oradiance.o radiance.c
2: gcc -I<Pfad zu den FORWISS-Headerdateien> -O3 -g3 -Wall -c
-fmessage-length=0 -ohelper.o helper.c
3: gcc -I<Pfad zu den FORWISS-Headerdateien> -O3 -g3 -Wall -c
-fmessage-length=0 -omain.o main.c
4: gcc -I<Pfad zu den FORWISS-Headerdateien> -O3 -g3 -Wall -c
-fmessage-length=0 -ointegration.o integration.c
5: gcc -I<Pfad zu den FORWISS-Headerdateien> -O3 -g3 -Wall -c
-fmessage-length=0 -obspline.o bspline.c
6: gcc -I<Pfad zu den FORWISS-Headerdateien> -O3 -g3 -Wall -c
-fmessage-length=0 -ocylinder.o cylinder.c
7: gcc -I<Pfad zu den FORWISS-Headerdateien> -O3 -g3 -Wall -c
-fmessage-length=0 -osensorarray.o sensorarray.c
8: gcc -L<Pfad zu den statischen FORWISS-Bibliotheken> -L<Pfad zur
GTK-Installation> -o<Name der ausführbaren Binärdatei>.exe
sensorarray.o radiance.o main.o integration.o helper.o cylinder.o
bspline.o -lgtk2gui -limageio -limages -latk-1.0 -lgdk-win32-2.0
-lglib-2.0 -lgobject-2.0 -lgtk-win32-2.0 -liconv -lpango-1.0 -lpng
-lbasics
```

5.3 Grafische Benutzeroberfläche

Zybesi sieht in seiner aktuellen Version lediglich eine grafische Benutzeroberfläche als Interaktionsmöglichkeit vor. Ein parametrisierter oder gescripteter Aufruf ist nicht möglich. Jedoch sollte es keinen all zu großen Aufwand machen, derartige Schnittstellen einzubauen.

Als einzige Ausgabemöglichkeit der Simulationsbilder wird dem Anwender im Moment nur die GUI selbst sowie ein PNG-Export angeboten. Ein automatisches Abspeichern von Simulationsbildern in beliebigen Formaten ist bisher nicht vorgesehen. Die grafische Benutzeroberfläche von Zybesi besteht aus drei Fenstern.

- Im ab dem Start sichtbaren Hauptfenster, kann man neben Form- und Lageparametern des Zylinders, Beleuchtungsparametern und Abstandsparametern weitere auf die Simulation Auswirkung habende Variablen setzen.
- Ein zweites, ebenfalls ab dem Start sichtbares Fenster, zeigt ausschließlich das berechnete Simulationsbild an.
- Das letzte Fenster dient der Angabe von Parametern für das Sensormaaß, wie die B-Spline-Koeffizienten oder der Grad des B-Splines, der das Sensormaaß approximiert. Es kann nur im realistischen Bildberechnungsmodus aufgerufen werden und ist beim Start von Zybesi nicht sichtbar.

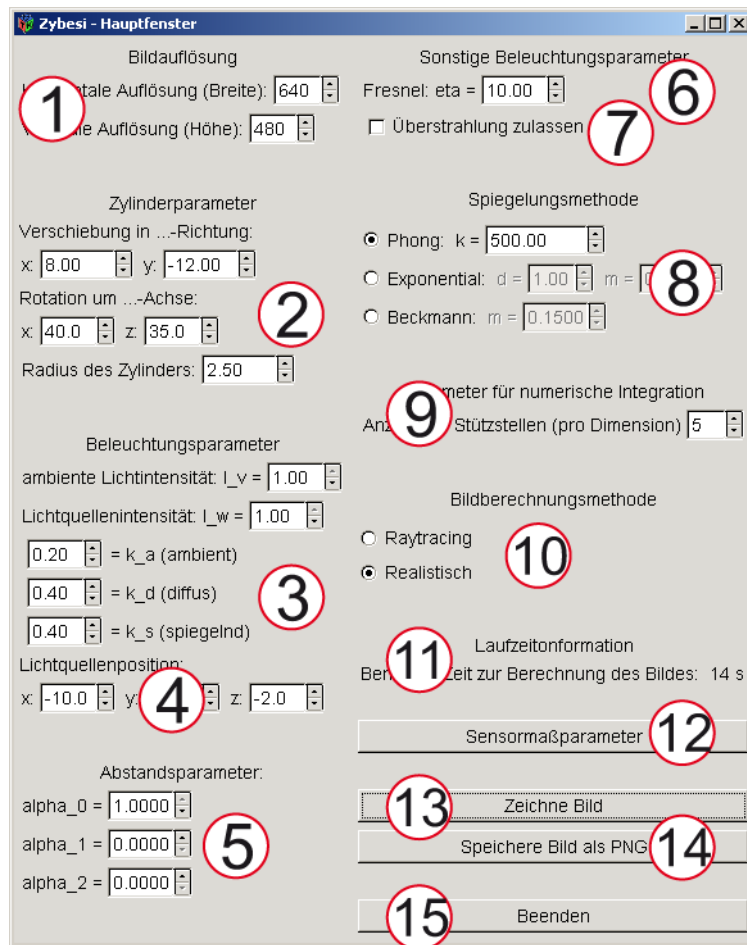


Abbildung 7: Das Hauptfenster von Zybesi.

1. Horizontale und vertikale Auflösung des zu erstellenden Simulationsbildes können hier spezifiziert werden.
2. In diesem Bereich lassen sich die Form- und Lageparameter d , α_1 , α_2 und r des Zylinders angeben, wie sie in 2.2 beschrieben werden.
3. Die für das Beleuchtungsmodell nach 2.4 benötigten Parameter I_v , I_w , k_a , k_d und k_s können an dieser Stelle eingegeben werden.
4. x -, y - und z -Koordinate der punktförmichen Lichtquelle nach 2.5 kann der Benutzer an dieser Stelle angeben.
5. Für die in Abschnitt 2.4.4 vorgestellte Formel zur Minderung der Lichtintensität nach dem quadratischen Entfernungsgesetz werden drei Parameter α_0 , α_1 und α_2 benötigt, die der Benutzer hier Zybesi übergeben kann.
6. Das Reflexionsgesetz nach Fresnel, das bei der Berechnung des spiegelnden Anteils verwendet wird, ist in Zybesi nur in einer sehr einfachen Version im-

plementiert (siehe 2.4.3). Vom Anwender kann der Parameter η im entsprechenden Textfeld eingegeben werden.

7. Wie in Abschnitt 4.1 bereits festgestellt wurde, garantieren die Formeln zur Berechnung des spiegelnden Anteils nicht, dass $I_s \in [0, 1]$. Durch Setzen des Häckchens erlaubt man die Verwendung von Werten > 1 für I_s . Anderfalls wird $\min\{1, I_s\}$ verwendet.
8. Zybesi unterstützt drei Varianten zur Berechnung des Parameters D , der in den spiegelnden Anteil einfließt. Neben der Methode von Phong und Beckmann kann der Benutzer auch eine exponentielle Methode wählen.
9. Zur numerischen Integration bei realistischer Bilderstellung verwendet Zybesi die in 2.9 präsentierte Simpson-Regel. Diese benötigt Stützstellen zur Annäherung des Integrals über einem Intervall, deren Anzahl hier spezifiziert werden kann.
Da die Simpson-Regel eine gerade Anzahl von Integrationsintervallen und somit eine ungerade Anzahl von Stützstellen benötigt, generiert Zybesi auch nur für eine ungerade Anzahl von Stützstellen Simulationsbilder. Zusätzlich arbeitet der realistische Bildberechnungsmodus mit mindestens 3 Stützstellen pro Dimension, also mit $3^2 = 9$ Stützstellen über einem Sensor.
10. Es stehen zwei Arten der Generierung des Simulationsbildes zur Verfügung: Neben einer zeitaufwändigeren realistischen Berechnungsmethode (siehe 3.2) unterstützt Zybesi auch Raytracing (siehe 3.1), welches zwar schneller, dafür aber realitätsferner ist.
11. Abschnitt 4.3 hat ja bereits gezeigt, dass der Bildgenerierungsprozess mitunter äußerst zeitaufwändig sein kann. Wie lange im konkreten Fall die Erstellung des Simulationsbildes gedauert hat, zeigt dieses Label an.
12. Ein Klick auf diesen Button öffnet das Fenster, mit dem Parameter für das Sensormaß verwaltet werden.
13. Wird diese Schaltfläche betätigt, beginnt der Berechnungsprozess des Simulationsbildes.
14. Nachdem ein Simulationsbild erstellt wurde, kann es als PNG-Datei abgespeichert werden, was jedoch nur in der Windowsversion von Zybesi möglich ist.
15. Button zum Beenden von Zybesi.

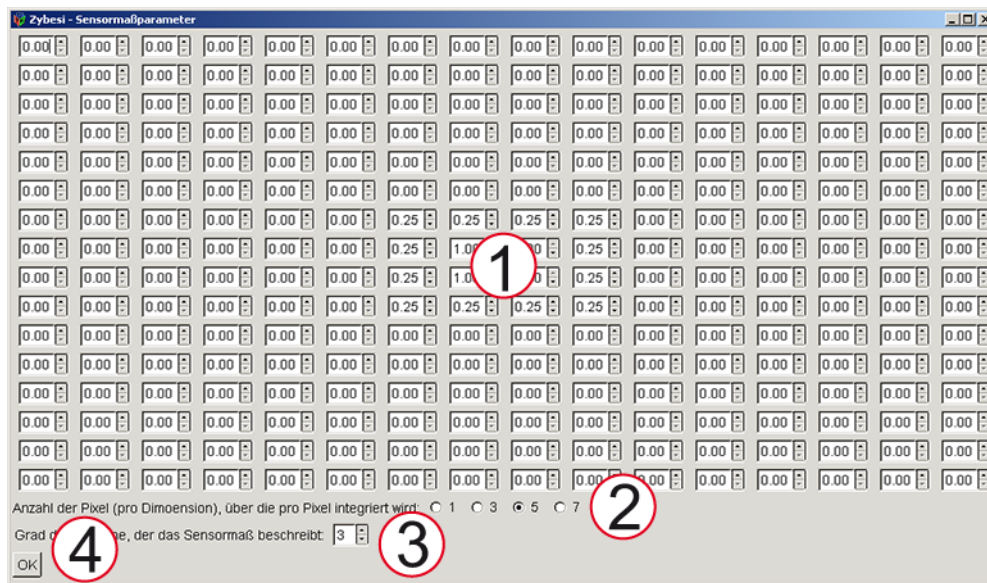


Abbildung 8: Fenster zur Eingabe von Parametern für das Sensormap.

1. 16×16 -Matrix zur Eingabe der B-Spline-Koeffizienten/Kontrollpunkte.
2. Auch benachbarte Bildpunkte haben Einfluss auf die Erstellung des Grauwertes eines Pixels. An dieser Stelle kann der Benutzer angeben, wie viele Sensoren (pro Dimension) bei der numerischen Integration zur Bestimmung der Lichtleistungsdichte beachtet werden.
3. Grad des B-Splines, der das Sensormap approximieren soll.
4. Das Klicken dieser Schaltfläche übernimmt die Einstellungen für das Sensormap.

6 Übersicht über die verwendeten Bezeichner

An dieser Stelle soll nochmal ein Überblick über alle in dieser Arbeit verwendeten Bezeichner gegeben werden, die auch in der Implementierung der Simulationssoftware Zybesi Verwendung finden.

$c = (0, 0, 0)$	Ursprung des Weltkoordinatensystems und optisches Zentrum der Lochkamera.
(u_1, u_2, u_3)	Basis des Raums, in der die Simulation stattfindet und Koordinatenachsen des Weltkoordinatensystems.
$\alpha_1 \in [0, 2\pi]$	Rotation der Zylinderhauptachse um die x -Achse des Weltkoordinatensystems. Die Rotation erfolgt dabei im Uhrzeigersinn, wenn man vom Ursprung aus in Richtung u_1 blickt.
$\alpha_2 \in [0, 2\pi]$	Rotation der Zylinderhauptachse um die z -Achse des Weltkoordinatensystems. Die Rotation erfolgt dabei im Uhrzeigersinn, wenn man vom Ursprung aus in Richtung u_3 blickt.
$d = (d_1, d_2, 0)$	Durchstoßpunkt der Zylinderhauptachse durch die xy -Ebene.
$f = (f_1, f_2, f_3)$	Richtung der Zylinderhauptachse nach der Rotation von $(0, 1, 0)$ um x - und z -Achse.
$a = (a_1, a_2, a_3)$	Normierter Richtungsvektor von einem Oberflächenpunkt zur Lochkamera. $a = c - x$.
$l = (l_1, l_2, l_3)$	Normierter Richtungsvektor von einem Oberflächenpunkt zur Lichtquelle.
$n = (n_1, n_2, n_3)$	Normalenvektor der senkrecht zur Objektoberfläche steht.
$h = (h_1, h_2, h_3)$	Halbvektor, sozusagen die „Winkelhalbierende“ zwischen a und l .
$b = (b_1, b_2, -1)$	Punkt in der Retina-/ $z = -1$ -Ebene. Wird in der Regel als Aufpunkt eines Sichtstrahls der Lochkamera durch c verwendet.
$s = (s_1, s_2, s_3)$	Richtung eines Sichtstrahls der Lochkamera. Sie ist definiert durch $c - b$.
$x = (x_1, x_2, x_3)$	Schnittpunkt eines Sichtstrahls der Lochkamera (Aufpunkt b mit Richtung s) mit dem durch d , α_1 , α_2 und r spezifizierten Zylinder. x ist also ein Oberflächenpunkt des Zylinders.

Tabelle 1: Übersichtstabelle über die im Programmierpraktikum verwendeten Bezeichner für Vektoren und Punkte.

$I_a \in [0, 1]$	Ambienter Anteil der Lichtmenge in einem Punkt oder über einer Fläche.
$I_d \in [0, 1]$	Diffuser Anteil der Lichtmenge in einem Punkt oder über einer Fläche.
$I_s \in [0, 1]$	Spiegelnder Anteil der Lichtmenge in einem Punkt oder über einer Fläche.
$I \in [0, 1]$	Beleuchtungsintensität in einem Punkt oder über einer Fläche.
$D \in [0, 1]$	Anteil der Microfacetten, die aufgrund ihrer Lage zur Spiegelung beitragen.
$G \in [0, 1]$	Geometrischer Abschwächungsfaktor. Beschreibt den Grad der gegenseitigen Verschattung der Facetten und der daraus resultierenden Absorption von Licht.
$F_\lambda \in [0, 1]$	Beschreibt das Reflexionsgesetz nach Fresnel unter Beachtung des Brechungsindex des Oberflächenmaterials.
C	Abschwächungsfaktor des Lichts bedingt durch den Abstand von Objekten nach dem quadratischen Entfernungsgesetz.
$tp_1, tp_2 \in \mathbb{R}^3$	Zwei Schattengrenzpunkte auf der Oberfläche des Zylinders, die von der Lochkamera gerade noch gesehen werden können und noch nicht zur „Rückseite“ des Zylinders gehören.

Tabelle 2: Übersichtstabelle über die im Programmierpraktikum verwendeten Bezeichner, die beim Beleuchtungsmodell benötigt werden.

7 Schluss

Das Programmierpraktikum, das diese Ausarbeitung zusammenfasst, lief über einen Zeitraum von etwa sechs Monaten. Hierbei geht es um die Erstellung der Software Zybesi, welche die Beleuchtung von Zylindern simuliert.

Dabei steht vor allem eine realitätsnahe Modellierung im Vordergrund. Aus diesem Grund wird zur Beleuchtungssimulation das Modell von Blinn verwendet. Zusätzlich wird das Verhalten von Digitalkameras berücksichtigt, indem die virtuelle Lochkamera eine Integration der Lichtleistungsdichte über der gesamten (und sogar benachbarten) Sensorfläche durchführt. Darüber hinaus wird auch das Sensormaß beachtet, denn die Sensoren der Kamera werten in der Regel nicht homogen über der gesamten Sensorappertur aus.

Literatur

- [1] Dr. Rainer Kickingeder: „Optische Vermessung partiell reflektierender Oberflächen“, Universität Passau, 2007.
- [2] Prof. Klaus Donner: Skript zur Vorlesung „Numerik“ im WS 2005/06, Universität Passau, 2005.
- [3] PD Georg Pisinger: Folien zur Vorlesung „Computergrafik“ im WS 2007/08, Universität Passau, 2007.
- [4] Homepage von Pov-Ray:
<http://www.povray.org/>
- [5] Homepage von FORWISS:
<http://www.forwiss.uni-passau.de/>
- [6] Homepage des GIMP-Toolkit:
<http://www.gtk.org/>
- [7] Homepage der GNU Compiler Collection:
<http://gcc.gnu.org/>
- [8] Homepage der Portierung von GCC auf Windows MinGW:
<http://www.mingw.org/>
- [9] Homepage von libpng:
<http://www.libpng.org/pub/png/libpng.html>