Implementierung und Vergleich verschiedener Kameramodelle im Ray Tracing Kontext

Studienarbeit Computervisualistik

Prof. Dr. Stefan Müller

Betreuer Markus Geimer

Koblenz, 04.02.2005

Valentina Hans

Inhaltsverzeichnis

1. Einführung.....4

2. Grundlagen.....6
   2.1 Raytracer.....6
   2.2 Dünne Linse.....8
   2.3 Dicke Linse.....9
   2.4 Monte-Carlo Integration.....11

3. Implementierung.....13

4. Vergleich.....19

5. Auswertung.....29

6. Literaturliste.....31

Abbildungsverzeichnis

1. Original Raytracer Bild ……...5
2.1 Diagramm: Raytracer…….....7
2.2 Dünne Linse…….....8
2.3.1 Dicke Linse……...9
2.3.2 Verbundlinse……...10
2.4.1 Monte-Carlo Punkte im Kreis……12
2.4.2 Monte-Carlo Dichte……12
3.1 Senden von Strahlen von der Linse aus…….....14
3.2 Diagramm 3: Erweiterter Raytracer…….....17
4.1 Dünne Linse R = 1, N = 15, Length = 10……...21
4.2 Dünne Linse R = 20, N = 15, Length = 10……21
4.3 Dünne Linse R = 11, N = 15, Length = 9..……22
4.4 Dünne Linse R = 11, N = 1, Length = 9..……...22
4.5 Dünne Linse, R = 11, N = 15, Length = 20……....23
4.6 Dünne Linse, R = 11, N = 15, Length = 15……....23
4.7 Dünne Linse, R = 11, N = 15,Length = 12…….……24
4.8 Dünne Linse, R = 11, N = 15,Length = 9……...24
4.9 Dünne Linse, R = 11, N = 15,Length = 6……...……....25
4.10 Dünne Linse, R = 11, N = 15,Length = 3……..……...…25
4.11 Dünne Linse, R = 11, N = 15,Length = 1……...……..26
4.12 Dünne Linse, R = 11, N = 15, Length = 8……....26
4.13 Dicke Linse, d = 2, R = 11, N = 15, Length = 8……..……27
4.14 Dicke Linse, d = 4, R = 11, N = 15, Length = 8……..27
4.15 Dünne Linse R = 11, N = 15, Length = 2……28

1. Einführung

Ein konventioneller Raytracer simuliert physikalisch korrekte Lichtverhältnisse einer Szene und sorgt für eine gute Approximation der Reflexionseigenschaften der Objektoberflächen. Dabei werden die optischen Eigenschaften einer realen Kamera vernachlässigt. Deshalb sehen von einem Raytracer gerenderte Szenen an jeder Stelle des Bildes scharf aus. Die Aufgabe dieser Arbeit ist die Simulation einer realen Linsenkamera, die Lochkamera des Raytracers ersetzt. Der Raytracer wird dabei um zwei Methoden erweitert, die der Dünnen und der Dicken Sammellinsen. Diese simulieren entweder die Arbeit einer Dünnen oder einer Dicken Linse. Die erweiterte Version soll die Fokussierung von einzelnen Objekten und die damit verbundene Unschärfe aller anderen Objekte in einer Szene ermöglichen.

Im zweiten Kapitel wird kurz die Funktionsweise des vorhandenen Raytracers und der Monte-Carlo Integration vorgestellt und die physikalischen Eigenschaften der Linsen beschrieben. Als nächstes wird auf die Implementierung der einzelnen Linsen eingegangen, die im darauf folgenden Kapitel anhand von Beispielbildern mehrerer Szenen verglichen werden. Zum Schluss werden die Ergebnisse dokumentiert und ausgewertet.

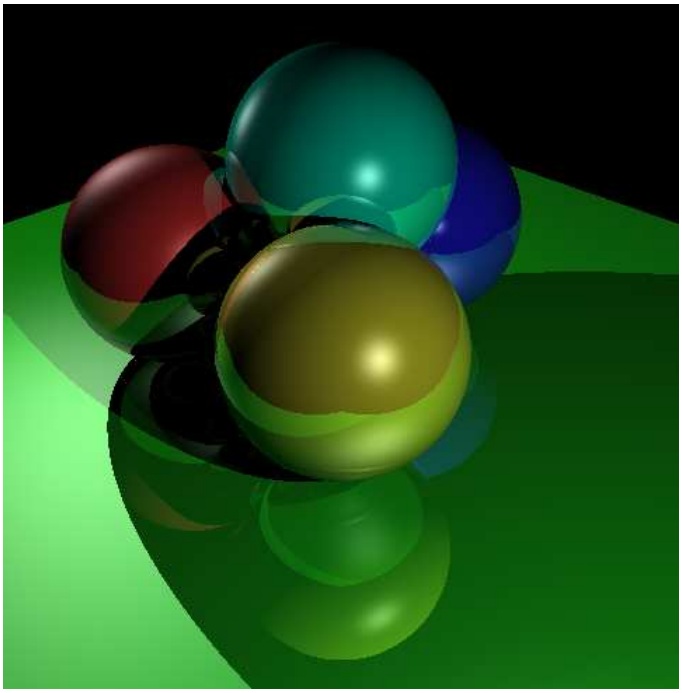

Abbildung 1: Original Raytracer Bild

2. Grundlagen

In diesem Kapitel werden die Grundlagen vorgestellt, die für das Weitere vorgehen benötig werden. 2.1 befasst sich mit der Funktionsweise des Raytracers. Darauf folgen zwei Unterkapitel mit physikalischen Eigenschaften der Dünnen und der Dicken Linsen. Abschließend wird die Bedeutung der Monte-Carlo Integration in dieser Arbeit erklärt.

2.1 Raytracer

Der mir zur Verfügung stehende Raytracer arbeitet wie eine Lochkamera und liefert ein, an jeder Stelle der Szene scharfes Bild (Abbildung 1). Um diesen Effekt zu erreichen ruft die Klasse `main` die Methode `renderToFile` der Klasse Raytracer auf, welche die Methode `trace` der gleichen Klasse aufruft (Diagramm 2.1). `Trace` schickt von der in `Scene` definierter Position der Camera aus Strahlen

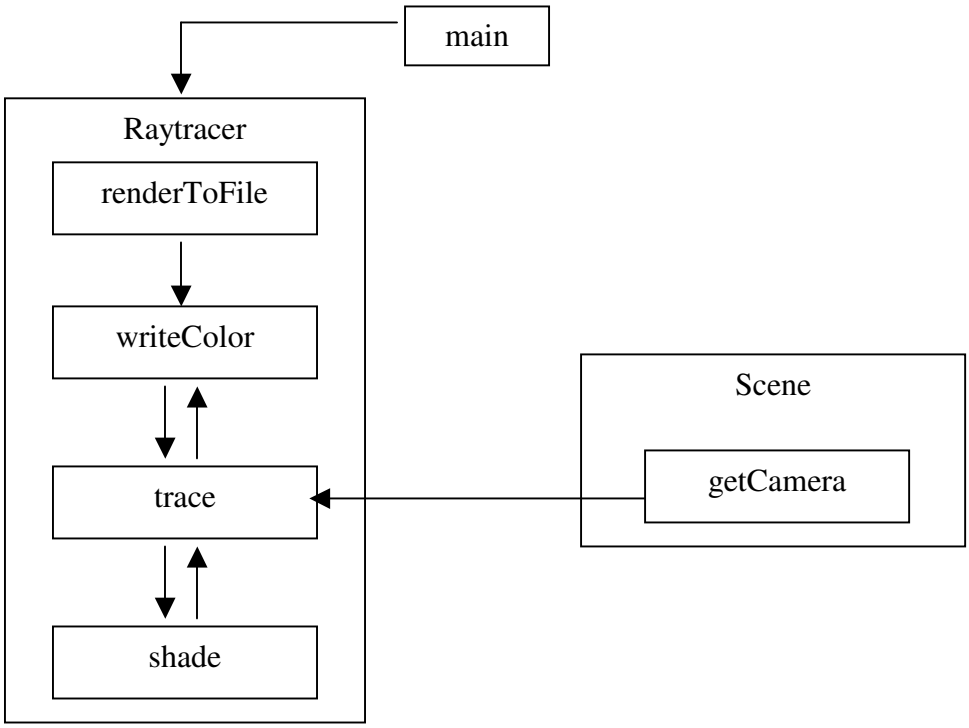

Diagramm 2.1: Raytracer

in die Szene, die die Szene abtasten und bei jedem Treffen auf ein Objekt die Farbe mit `shade` an dem Schnittpunkt des Objektes mit dem Strahl bestimmen. Wenn ein Strahl nach dem Erreichen der maximalen Strahltiefe `maxdepth` noch kein Objekt getroffen hat, wird die Hintergrundfarbe schwarz in die Ausgabedatei mit `writeColor` gespeichert. Die Szene sowie Szenenobjekte `Triangle` und `Sphere` und deren Farbe sind in der `main`-Klasse definiert. Alle Punkte und Farben werden als 3D Vektoren von `float`-Werten dargestellt.

2.2 Dünne Linse (Thin Lens)

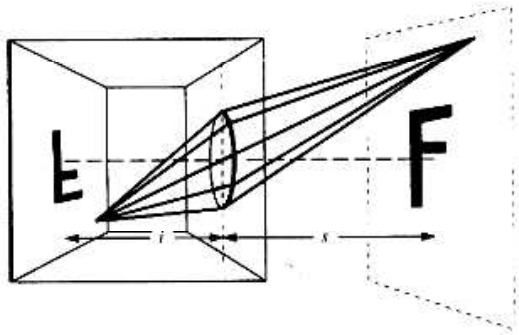

Abbildung 2.2: Dünne Linse

In einem Lochkameramodell wird ein an jeder Stelle scharfes Bild erzeugt. Im Gegensatz dazu wird bei den Linsenmodellen nur das Objekt, das in der Fokusweite **s** liegt, scharf dargestellt. Das Licht wird von jedem Punkt des Objekts durch die Linse gebrochen und auf der anderen Seite der Linse in dem Pixel, an der dem Ursprungspunkt entsprechenden gespiegelten Stelle in der Entfernung **i** auf der Bildebene gebündelt (Abbildung 2.2). Die Beziehung der Fokusweite **s**, der Brennweite **f** und der Bildweite **i** zueinander wird als Abbildungsgleichung bezeichnet:

$$1/s + 1/i = 1/f \qquad [1]$$

Da eine Dünne Linse nur eine Hauptebene hat, an der jeder Strahl einmal gebrochen wird, kann die Dicke der Dünnen Linse vernachlässigt und die Linse

zur Vereinfachung als eine flache Scheibe mit dem Radius **R** angenommen werden. Aber bei einer realen Kamera wird der Radius der Linse durch den Öffnungswinkel der Kamera beschränkt, deshalb wird eine weitere Variable **f-number (mR)** für den Öffnungswinkel eingeführt und aus dem Wert dem Radius berechnet:

$$mR = f / (2 * R). \qquad [2]$$

2.3 Dicke Linse (Thick Lens)

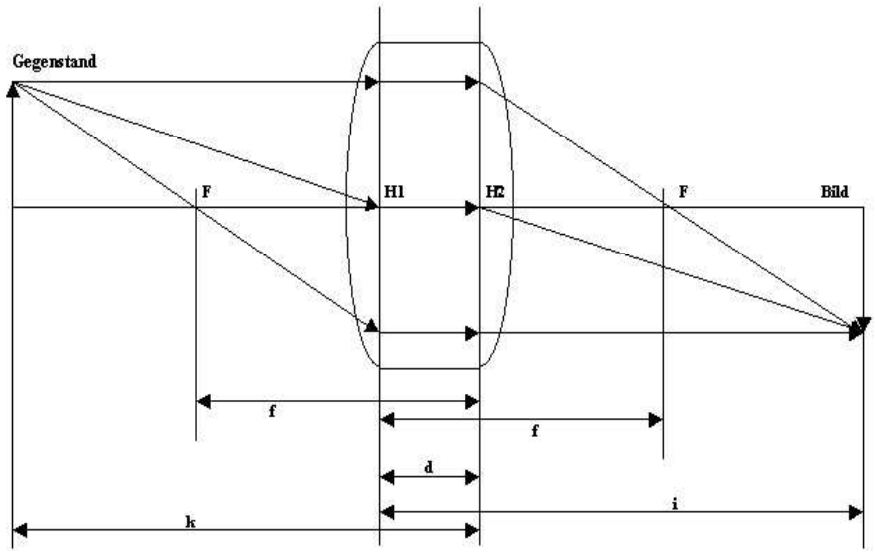

Abbildung 2.3.1: Dicke Linse

Eine dicke Linse hat im Gegensatz zu der dünnen Linse zwei Hauptebenen **H1** und **H2** an denen der Strahl gebrochen wird (Abbildung 2.3.1). Alle Messungen

von Entfernungen müssen von der weiter entfernt liegenden Hauptebene aus gemacht werden. Die Brennweite wird dann, von links nach rechts gesehen, von der linken Hauptebene H1 über die Dicke der Linse bis zu dem Brennpunkt gemessen. Die Bildweite verläuft von der ersten Hauptebene **H1** über die Linse bis zu der Bildebene und die Gegenstandsweite von dem Gegenstand bis zu der zweiten Hauptebene **H2**. Die Dicke der Linse **d** wird als die Entfernung zwischen der ersten und der zweiten Hauptebene betrachtet und muss bei der Verwendung der Fokusweite **s** und der Brennweite **f** berücksichtigt werden. Jeder einzelne Strahl wird an zwei Stellen gebrochen, an der ersten und zweiten Hauptebene. Zwischen den Hauptebenen wird der Strahl parallel verschoben. Nach dem zweiten Brechen verläuft er wie bei der dünnen Linse bis zum Zielpunkt auf der Bildebene weiter.

Nach dem gleichen Prinzip funktionieren auch zwei dünne hintereinander stehende Linsen. Eine Anordnung von mehreren gleichen oder eine Kombination aus Dünnen und Dicken Linsen hintereinander, genannt Verbundlinse,
wird physikalisch auch als eine Dicke Linse betrachtet (Abbildung 2.3.2). In diesen Fall werden die Strahlen mehr als zweimal gebrochen. Da aber die Brechungen im Verlauf der Verbundlinse ausgeglichen werden, gelten sie als zwei Dünne Linsen hintereinander. Zur Vereinfachung wird die Linse als zwei

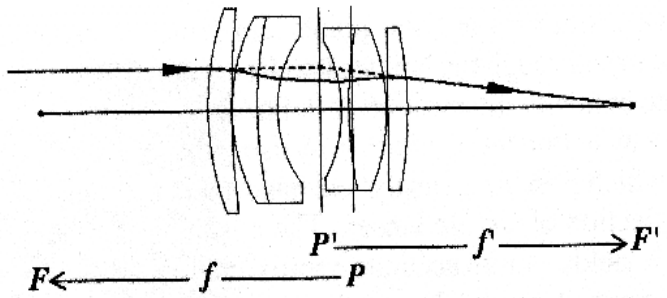

Abbildung 2.3.2: Verbundlinse

hintereinander stehende Kreisscheiben angenommen. Die Strahlen werden an jeder der Linsen einmal gebrochen.

Der einzige Unterschied zu der dünnen Linse ist, dass bei der Implementierung die Dicke der Linse **d** von der Fokusweite, der Bildweite und der Brennweite abgezogen werden muss. Danach wird genau so wie bei der Dünnen Linse das Licht hinter der Bildebene in der Entfernung **i** gebündelt.

2.4 Monte-Carlo Integration

Monte-Carlo Integration wird verwendet, um bestimmte Integrale mit Hilfe von Zufallszahlen in einer Dichte im zweidimensionalen Raum zu berechnen. Dazu wird aus einer 2D- Dichtefunktion eine 2D Verteilungsfunktion bestimmt und nach den Unbekannten aufgelöst. Bei einer Linse kann die Monte-Carlo-Methode zur Bestimmung von gleichverteilten 2D-Zufallspunkten auf der kreisförmigen Linse verwendet werden.

Als erstes werden zwei Zufallswerte **p1** und **p2** zwischen 0 und 1 mit Hilfe eines einfachen Zufallsgenerators `(rand())` bestimmt. Aus diesen Zufallszahlen und dem Radius **mR** werden dann die Entfernung **r** und der Winkel **p** zum gesuchten Punkt auf der Kreisscheibe (Abbildung 2.4.2) ermittelt:

$$r = mR * \mathrm{sqrt}\,(p1) \qquad [3]$$

$$p = 2 * PI * p2, \qquad [4]$$

und anschließend die Koordinaten im x-y-Koordinatensystem (Abbildung 2.4.1) berechnet als:

$$x = r * \cos\,(p) \qquad [5]$$

$$y = r * \sin\,(p) \qquad [6]$$

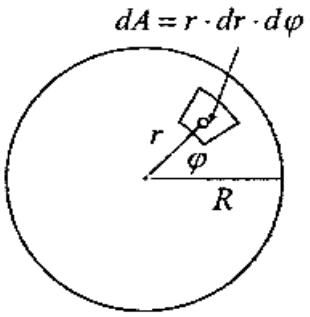

Abbildung 2.4.1: MonteCarlo Punkte im Kreis

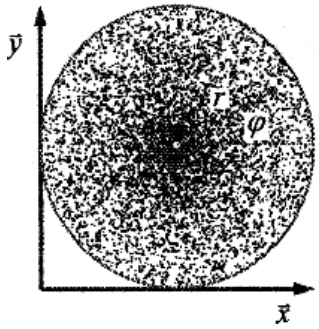

Abbildung 2.4.2: MonteCarlo Dichte

3. Implementierung

Der nachfolgende Kapitel befasst sich mit der Implementierung der Monte-Carlo Integration und der Linsen, die im vorherigen Kapitel vorgestellt wurden, in den Raytracer.

Um eine reale Kamera simulieren und den Fokussierungseffekt erreichen zu können, müssen Strahlen aus mehreren, mit Hilfe von MonteCarlo zufällig gewählten Punkten auf der Linse zu jedem Punkt der Bildebene geschickt werden. Nachdem ein Strahl seinen Zielpixel erreicht hat, setzt er seinen Weg in der Szene fort. Sobald er auf ein Objekt trifft, wird die Farbe am Schnittpunkt mit dem Objekt bestimmt und auf die Bildebene projiziert. Weil es zu umständlich ist die Szene auf einer Seite vor die Linse in Blickrichtung der Kamera und die Bildebene auf dem Kopf stehend hinter der Linse zu implementieren, wird die Bildebene vor die Szene in der Fokusweite **i** parallel zur der Linse platziert. Die Bildebene befindet sich somit auf der gleichen Seite der Linse wie die Szene (Abbildung 3.1). Aus diesem Grund werden die Fokusweite und die Bildweite durch nur einen Wert **s** definiert. Bei der Implementierung verwende ich statt **s** die Variable `Length`.
Der Öffnungswinkel der Kamera wird als:

```
mR = f / (2 * R)
```

aus dem Radius **R** und der Brennweite der Linse **f** ausgerechnet. Der neue Wert wird in der Implementierung als Radius der Linse verwendet.

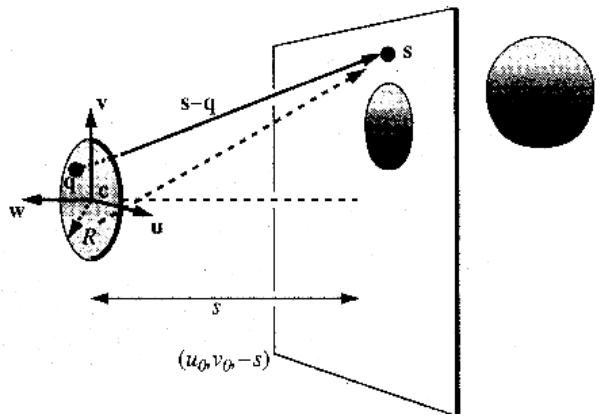

Abbildung 3.1: Senden von Strahlen von der Linse aus

Bei der nachfolgenden Beschreibung fange ich am tiefsten Punkt der Hierarchie, mit den Punkten auf der Linse an. Am Ende des Kapitels werde ich kurz beschreiben wie das Projekt tatsächlich durchlaufen wird.

Als erstes werden die Zufallswerte auf der Linsenfläche mit Hilfe der Monte-Carlo Integration in der Methode `getMonteCarlo(R)` der Klasse `MonteCarlo` erzeugt. Dafür werden zwei mit einem einfachen Zufallsgenerator `rand()` ausgewählte Werte zwischen 0 und 1 in die nach den zwei unbekannten aufgelöste 2D-Verteilungsfunktion eingesetzt.

```
float v = (float)rand() / RAND_MAX;
float w = (float)rand() / RAND_MAX;
float r = float(R) sqrt(v);
float p = 2 * M_PI * w;
```

Die ausgerechneten Werte werden mit

```
float x = r * cos(p);
float y = r * cos(p);
```

in das kartesische Koordinatensystem transformiert. Um vollständige 3D Punkte auf der Linse zu erhalten, wird der z-Wert auf null gesetzt.

```
origin =  Vector (x, y, z);
     return origin;
```

Die Punkte auf der Linse müssen jetzt in das Kamerakoordinatensystem so verschoben werden, dass sie in der `up-u`-Ebene liegen (in Abbildung 3.1 v-u-Ebene, `lookAt` ist –w-Achse), die zu der Bildebene parallel liegt.

```
Vector c = camera.getPosition();
Vector lookAt = camera.getLookAt();
Vector up = camera.getUp();

// Dritter Vektor der Kamera, steht
   senkrecht auf den up- und
lookAt-Achsen
Vector u = (cross(up, lookAt));

MonteCarlo m;
origin = m.getMonteCarlo(R);

origin = c + origin[X] * u + origin[Y] * up;

    return origin;
```

Von den `origin` Punkten aus werden Strahlen zu jedem Pixel auf der Bildebene geschickt. Die Zielpunkte `goalpoint` werden in einer Schleife über das Bild ermittelt und zum Ausrechnen der Richtung des Strahls, die zusammen mit origin einen Strahl definiert, verwendet.

```
direction = goalpoint - origin;
ray = Ray(mOrigin, mDirection);
```

Ray wird von der Methode `trace` in die Szene geschickt, bis er auf ein Objekt trifft `(closestIntersection)`. An dem Schnittpunkt des Strahls mit dem Objekt wird immer noch von `trace` aus mit der Methode `shade` die Farbe bestimmt und im Bild gespeichert.

```
color = trace(ray, 1);
```

Für jeden Punkt auf der Bildebene `goalpoint` werden in N Schleifendurchläufen Zufallspunkte `origin` auf der Linse erzeugt und Farbwerte `color` bestimmt. Color wird für alle Linsenpunkte zu einem `goalpoint` aufsummiert und in `image[x][y]` gespeichert.

```
image[x][y] = image[x][y] + color;
```

Am Ende werden die Farbwerte in `image[x][y]` jeweils durch die Anzahl der Zufallspunkte N geteilt und in der Ausgabedatei gespeichert.

```
image[x][y] = image[x][y] / N;
writeColor(fp, image[x][y]);
```

Wenn ein Strahl, der in die Szene geschickt wurde, auf kein Objekt getroffen hat, nachdem er die maximale Strahltiefe `maxdepth` erreicht hat, wird die Farbe schwarz gespeichert.

Die Implementierung einer Dicken Linse besitzt nur wenige Unterschiede zu der Dünnen. Bei der Dicken wird von der Fokusweite `Length` die Dicke der Linse **d** abgezogen und hat über die aus Length ausgerechneten Brennweite `FocalLength` Auswirkung auf den Radius **mR** [2].

```
float Length = Length - d;
```

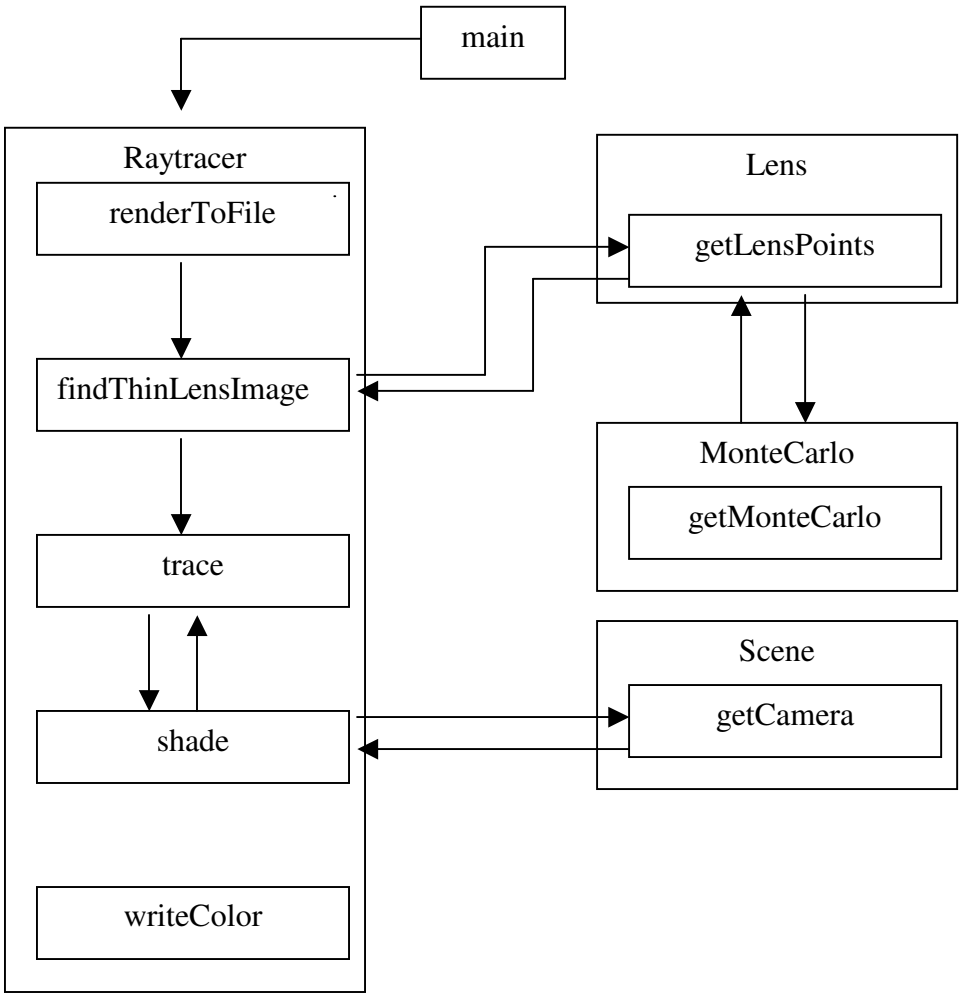

Diagramm 3.2: Erweiterter Raytracer am Beispiel der Dünnen Linse

Beim Starten des Projekts ruft `main` die Methode `renderToFile` in Raytracer auf, die wiederum `findThinLensImage` aufruft (Diagramm 3.2). Bei `findThinLensImage` wird in einer Schleife über die Bildebene gegangen und der jeweilige Zielpunkt der Strahlen bestimmt. In dieser Schleife ist noch eine, in der über die N zufällige Ausgangspunkte auf der Linse zu jedem in vorhergehenden Schleifen bestimmten Zielpunkt auf der Bildebene gegangen wird. Hier wird die Klasse `Lens` aufgerufen, die von MonteCarlo Zufallspunkte auf der Linse bekommt. Zurück in `Lens` wird jeder der Punkte auf der Linse zu der Kamera transformiert und an `findThinLensImage` übergeben. Von diesen Punkten aus gehen Strahlen zu dem aktuellen Punkt auf der Bildebene

`(trace(ray, 1))`. Sobald ein Strahl auf ein Objekt trifft, wird an dem Schnittpunkt des Strahls mit dem Objekt die Farbe bestimmt und für alle N Zufallspunke pro Zielpunkt aufaddiert und später durch N geteilt.
Eine Dünne Linse wird mit 4 Parametern aufgerufen: dem Dateinamen, dem Radius R, der Anzahl der Monte-Carlo-Punkte auf der Linse N und der Entfernung des Objektes im Fokus `Length`. Bei der Dicken Linse kommt noch die Dicke der Linse d.

Die Szenendaten werden in nff-Dateien gespeichert und vom Raytracer aufgerufen. An erster Stelle stehen immer die Kameradaten gefolgt von der Lichtquelle und den Szenenobjekten.

4. Vergleich

Nachdem dar erweiterte Raytracer zum laufen gebracht wurde, erfolgte das Testen der Arbeit anhand verschiedener Szenen mit unterschiedlichen Parametern.

Beim starten des Raytracers hat man die Wahl zwischen der Dünnen und der Dicken Linse.
Diese werden mit 3, bei der Dicken Linse mit 4 Werten aufgerufen und können bei allen Werten außer 0 Ergebnisse liefern. Aber um gute Werte, die eine reale Kamera simulieren, zu erhalten, wurden für jede Variable Intervalle in denen diese liegen sollten gefunden.
Als ersten Wert bestimmt man die Größe der Linse R. Bei einem zu kleinen Radius ist das Ergebnisbild auch bei einer optimalen Fokusweite zu verschwommen (Abbildungen 4.1). Bei einem zu großen Radius - immer scharf (Abbildung 4.2). Die besten Ergebnisse erzielt man mit den Werten zwischen 5 und 14. Innerhalb diesen Intervalls gilt: je größer der Radius, desto besser sind die Ergebnisse.
Der nächste Wert den man wählt, ist die Anzahl der zufälligen Punkte auf der Linse N, die zu jedem Pixel auf der Bildebene gesendet werden. Bei 10 Strahlen pro Pixel und mehr, erreicht man die besten Ergebnisse (Abbildungen 4.3 und 4.4). Da aber der Raytracer sehr langsam ist, ist es besser die Werte von N so niedrig wie möglich zu halten.
Der letzte und für die Fokussierung wichtigste Wert ist die Entfernung des Objekts im Fokus beziehungsweise der Bildebene von der Linse (`Length`). Die Entfernung verläuft von dem Ursprung der Kamera aus entlang der optischen Achse (Blickrichtung `lookAt`). Bei `Length` kommen nur Werte in Frage, die zwischen den Positionen des vordersten Objekts und des von der Kamera am

weitesten entfernten Objekts liegen. Wenn `Length` größer (Abbildung 4.5) oder kleiner (Abbildung 4.11) ist, sind alle Objekte im Bild unscharf. Denn in der Umgebung existieren keine Kanten, an denen die Zerstreuung der Farbwerte hätte beobachtet werden können. Der Hintergrund ist immer gleichmäßig schwarz. Die Vergrößerung der Bildweite lässt sich sehr gut an der Dünne Linse Bildfolge 4.5 bis 4.11 beobachten. In diesen Bildern wird die Fokusweite von 20 bis 1 in dreier Schritten verringert.

Bei einer Dicken Linse kann zusätzlich **d**, die Dicke der Linse variieren. Der einzigen bemerkbare Unterschiede zwischen der Dünnen und der Dicken Linse sind die Entfernung `Length` des Objekts im Fokus von der Linse und die Auswirkung der Dicke der Linse auf den Öffnungswinkel der Kamera **mR**. Sie macht den Radius etwas kleiner [2]. Nach mehrmaligem Ausführen von dünnen und dicken Linsen an Szenen mit gleichen Parametern, wurde festgestellt, dass die Abweichungen bei kleinem **d** sehr gering und kaum erkennbar sind (Abbildung 4.12 und 4.13). Bei einem größeren **d**-Wert ist der Unterschied besser erkennbar (Abbildung 4.14), lässt sich aber bei Verkleinerung von `Length` relativ gut ausgleichen (Abbildung 4.15). Bei der Wahl von **d** ist zu beachten, dass diese nie größer als `Length` sein darf. In einem solchen Fall wird `Length` negativ und das Ausgabebild wird schwarz. Wegen der Ähnlichkeit der Ergebnisse der beiden Linsenformen, kann durch eine Dünne mit einer um die Dicke der Linse verringerten Fokusweite ersetzt werden.

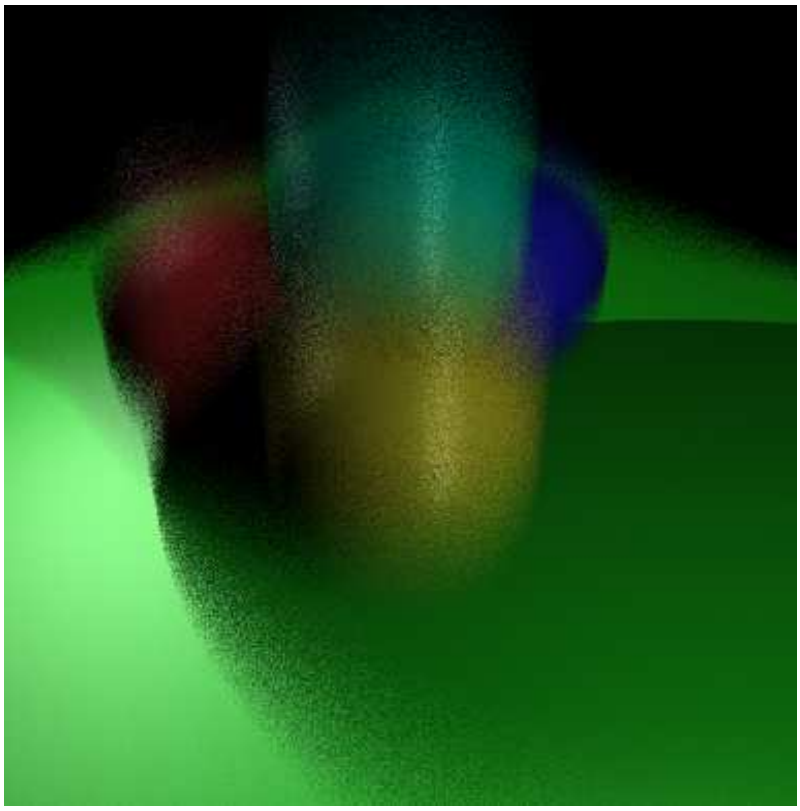

Abbildung 4.1: Dünne Linse R = 1, N = 15, Length = 10

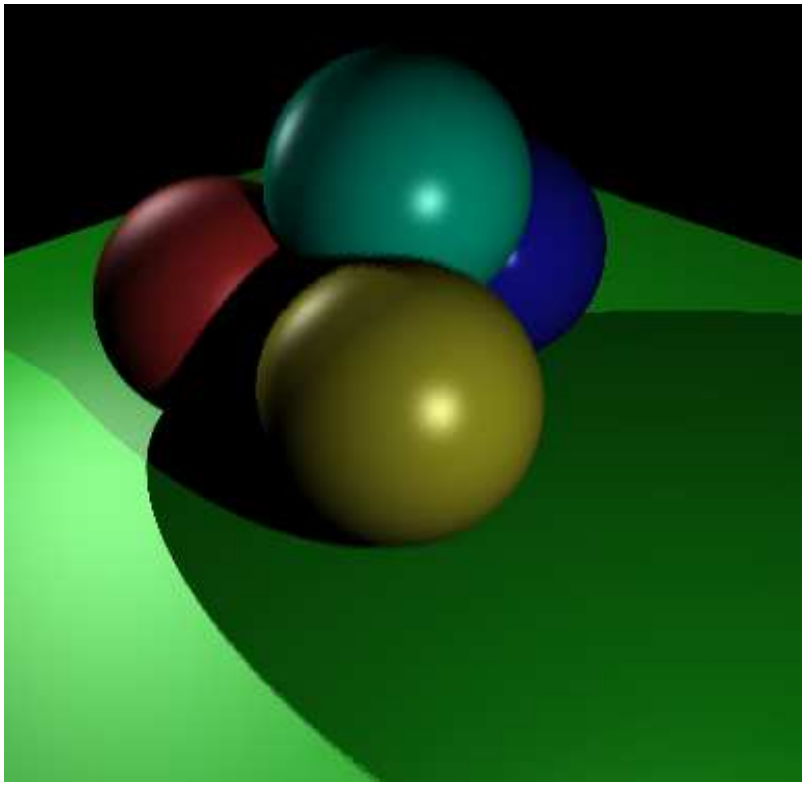

Abbildung 4.2: Dünne Linse, R = 20, N = 15 Length = 10

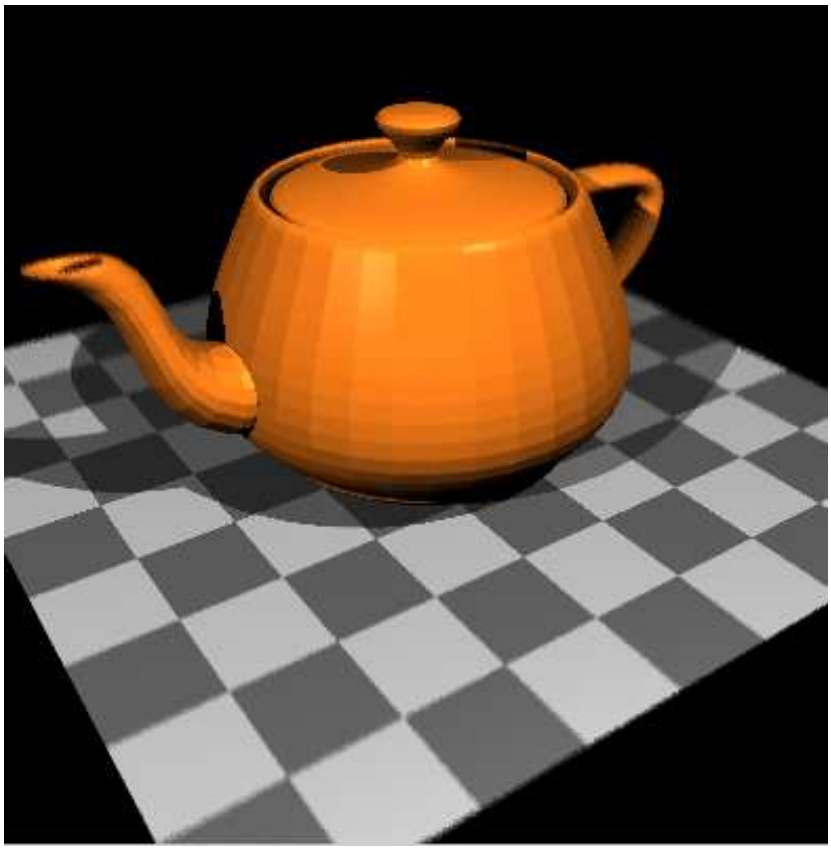

Abbildung 4.3: Dünne Linse, R = 11, N = 15, Length = 9

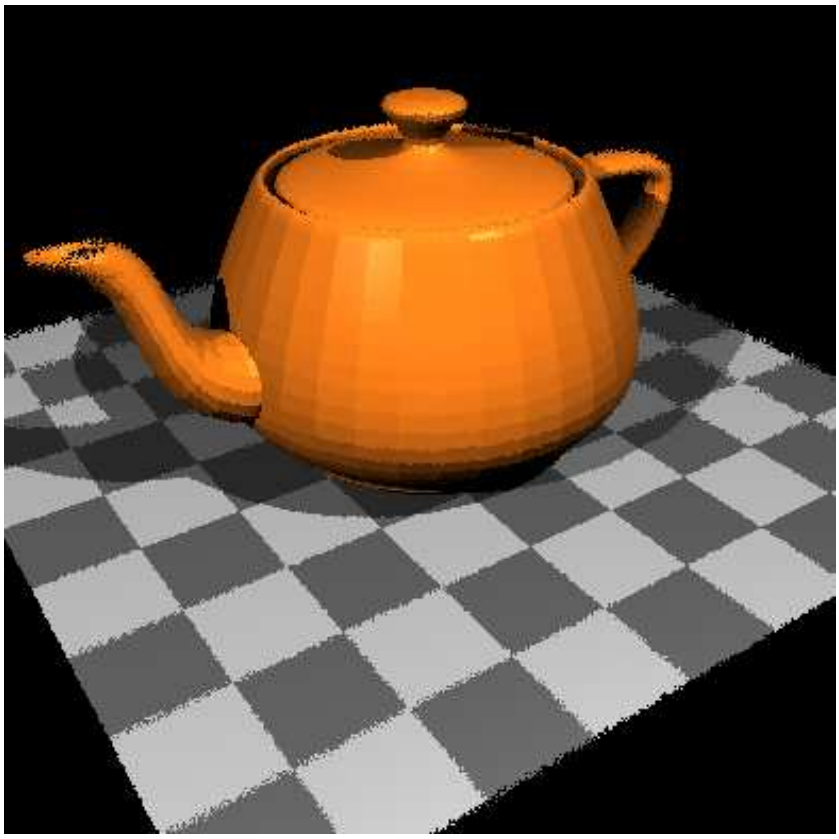

Abbildung 4.4: Dünne Linse, R = 11, N = 1, Length = 9

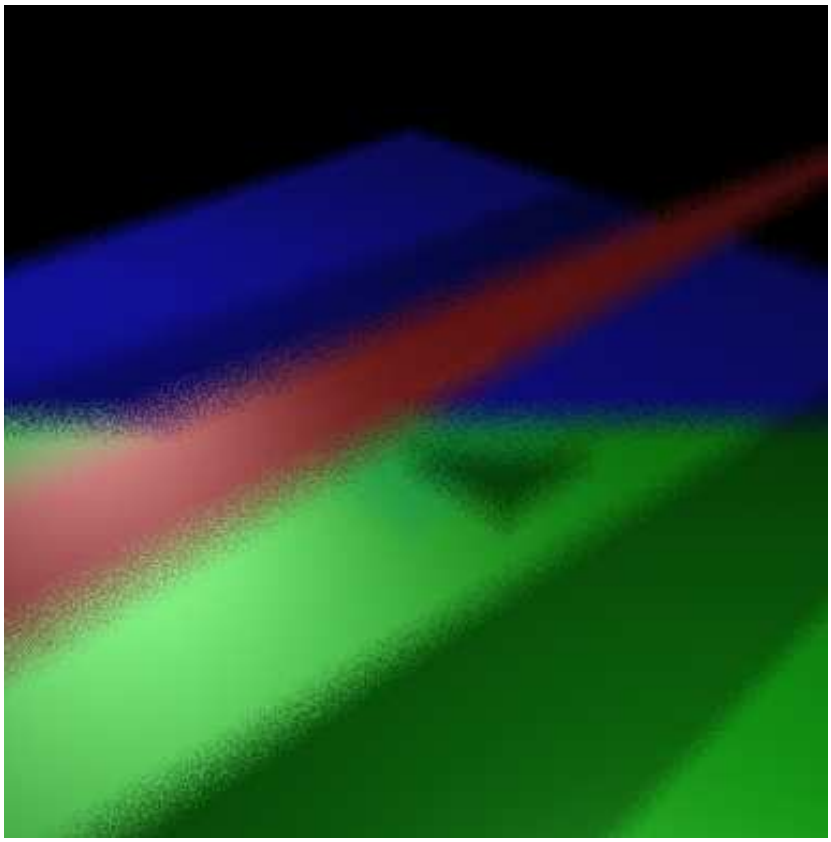

Abbildung 4.5: Dünne Linse, R = 11, N = 15, Length = 20

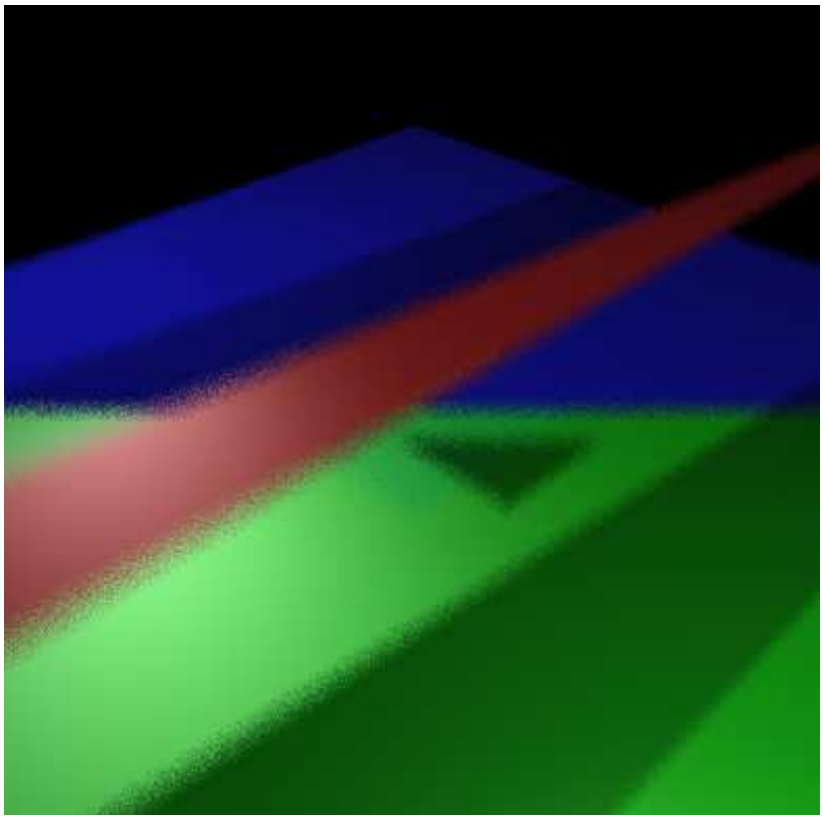

Abbildung 4.6: Dünne Linse R = 11, N = 15, Length = 15

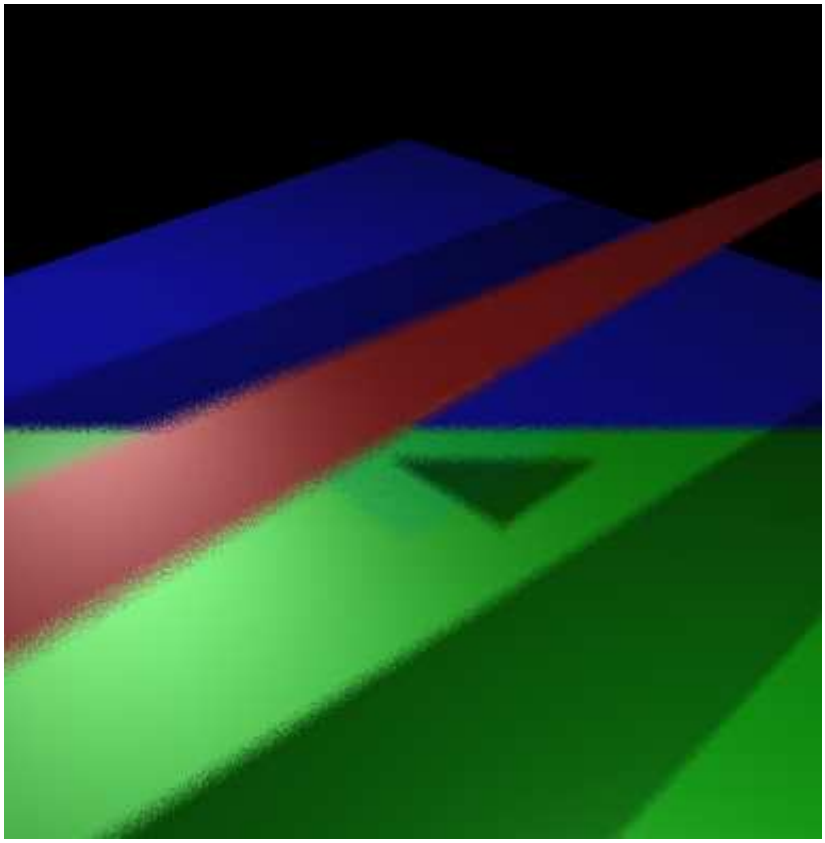

Abbildung 4.7: 12 Dünne Linse, R = 11, N = 15, Length = 12

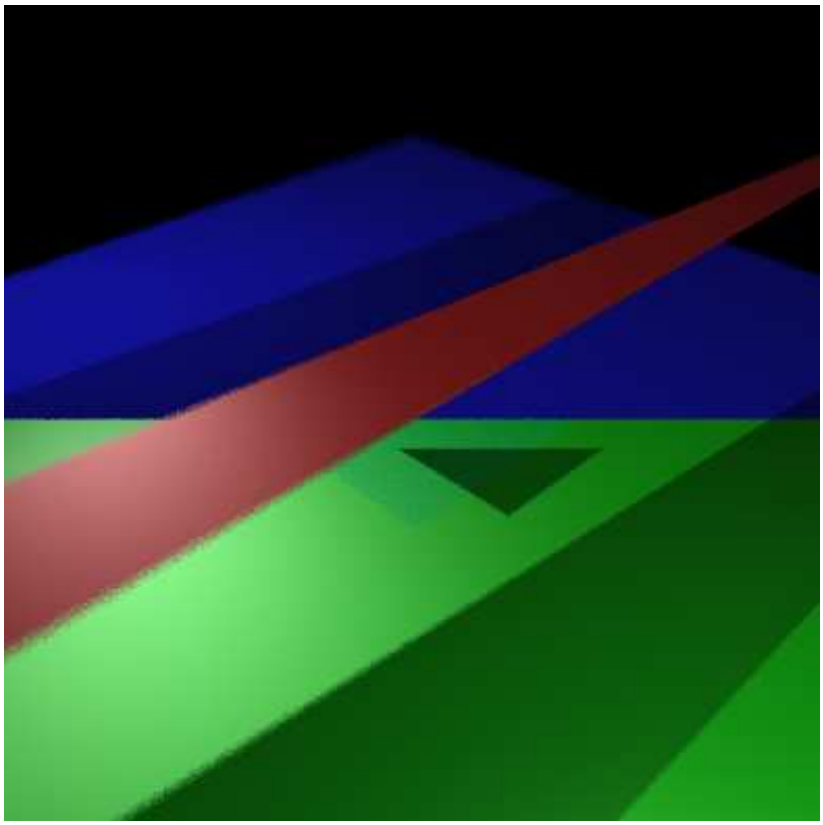

Abbildung 4.8: Dünne Linse, R = 11, N = 15, Length = 9

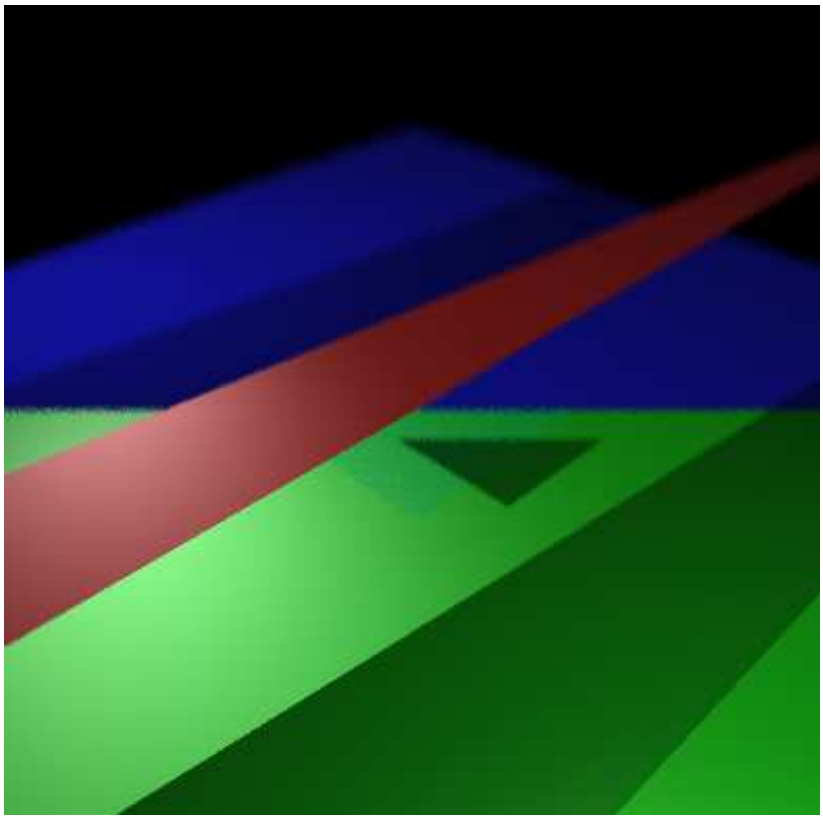

Abbildung 4.9: Dünne Linse, R = 11,N = 15, Length = 6

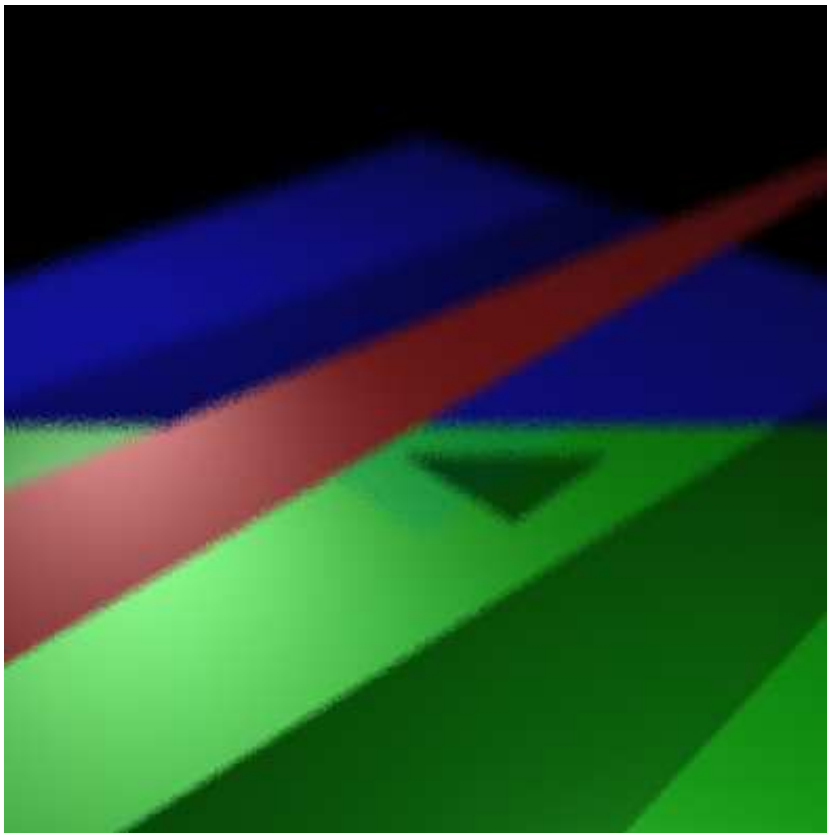

Abbildung 4.10: Dünne Linse, R = 11,N = 15, Length = 3

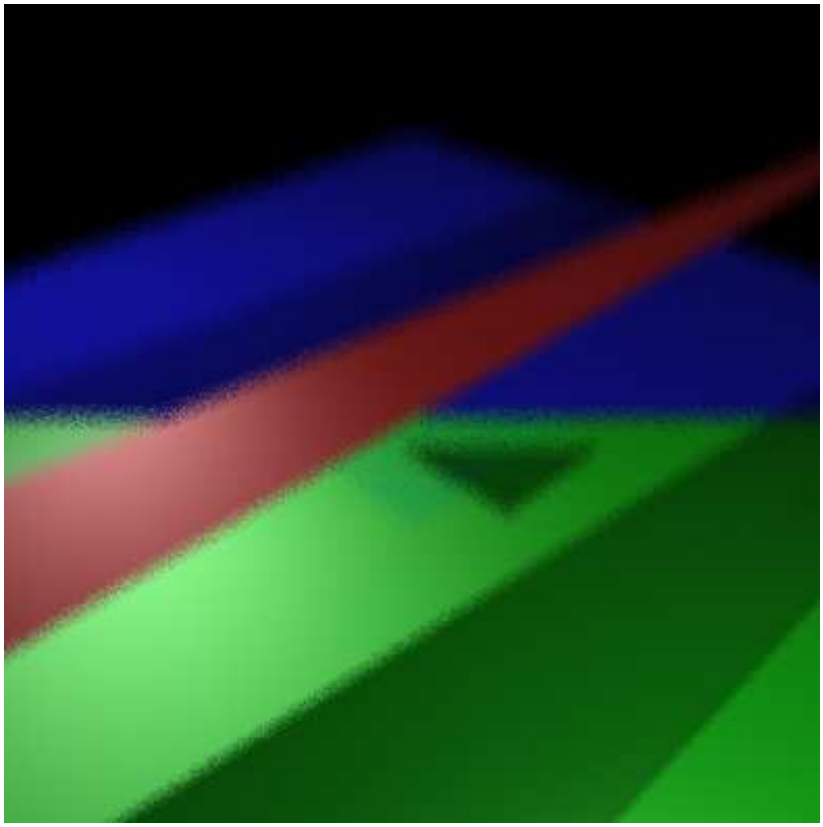

Abbildung 4.11: Dünne Linse, R = 11, N = 15, Length =1

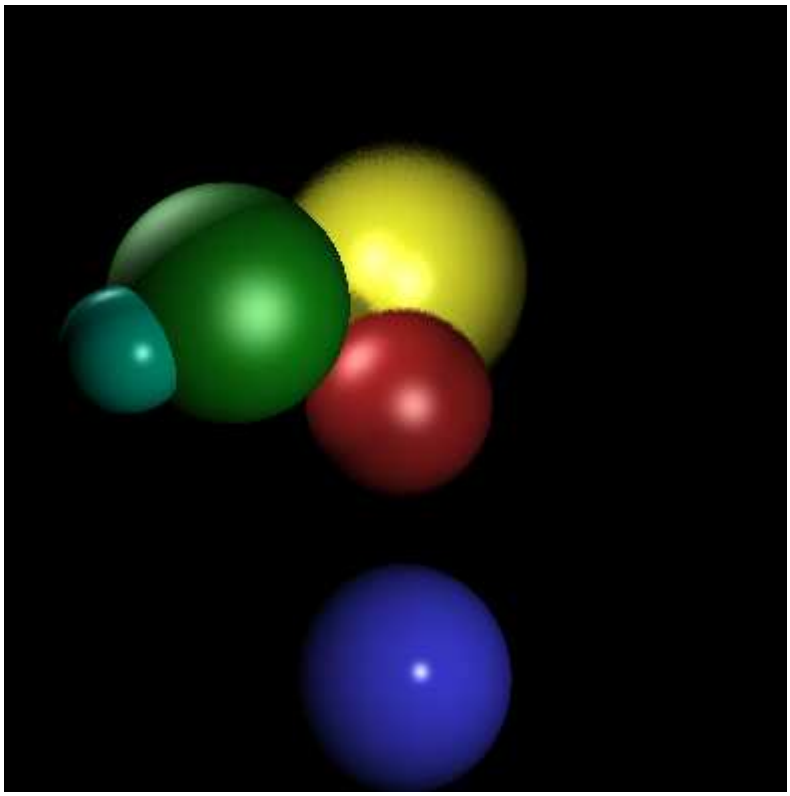

Abbildung 4.12: Dünne Linse, R = 11, N = 15, Length = 8

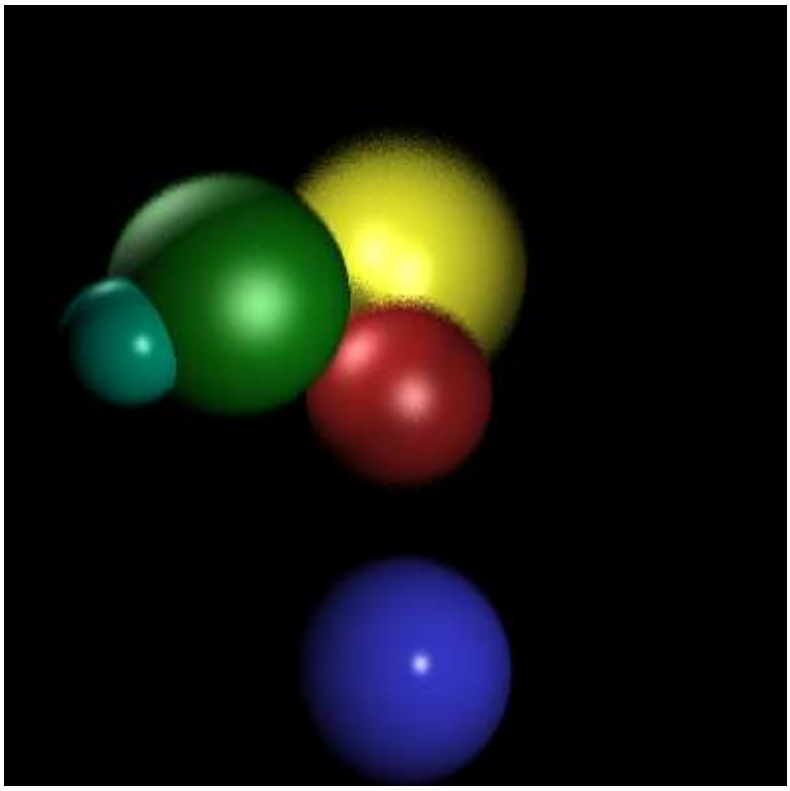

Abbildung 4.13: Dicke Linse, d = 2, R = 11, N = 15, Length = 8

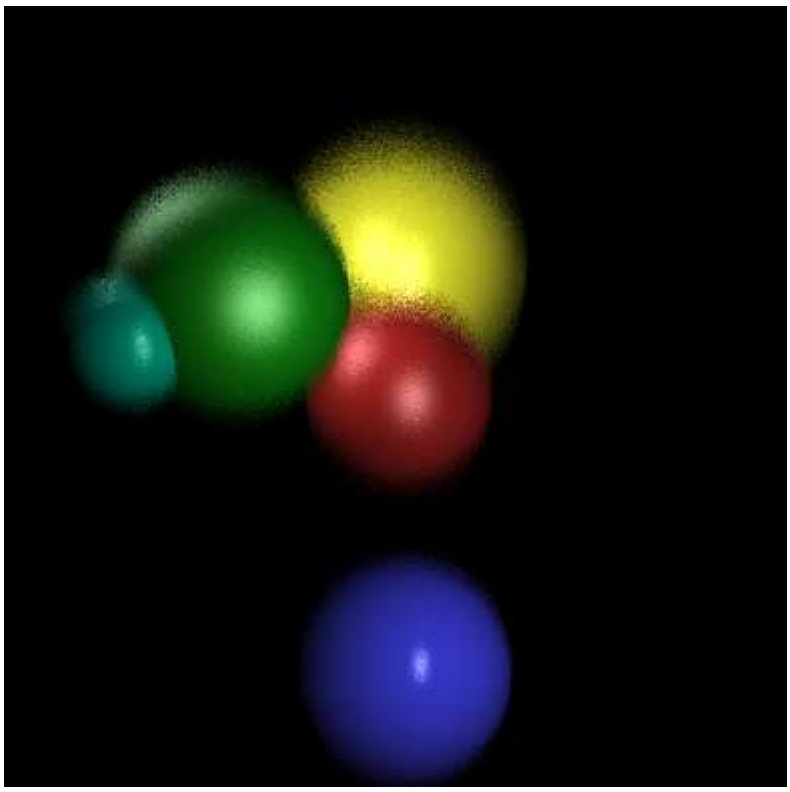

Abbildung 4.14: Dicke Linse, d = 4, R = 11, N = 15, Length = 8

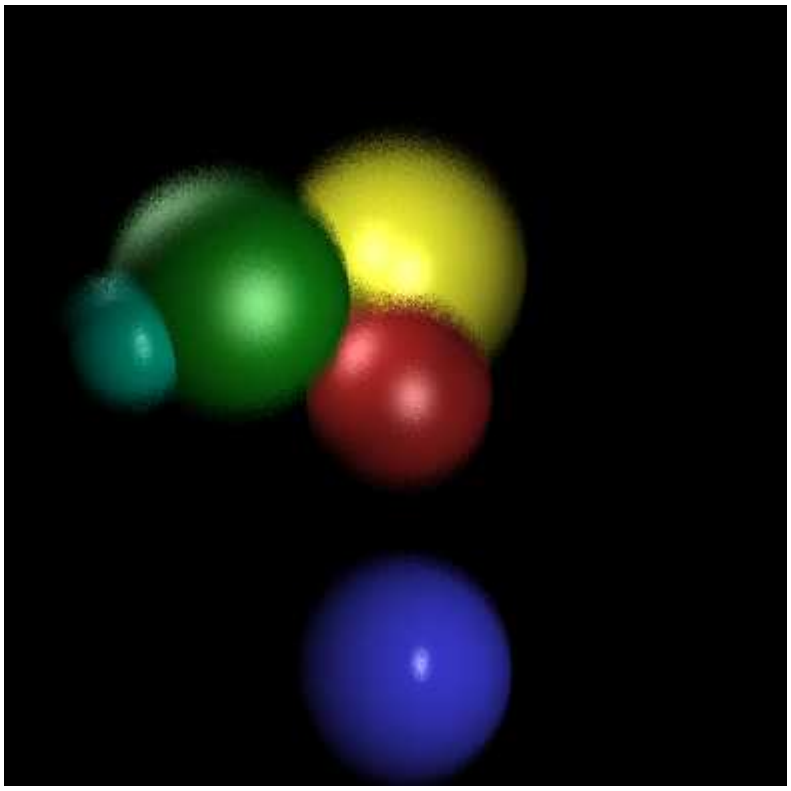

Abbildung 4.3: Dünne Linse, R = 11, N = 15, Length = 2

5. Auswertung und Fazit

Der Effekt der Fokussierung wird dadurch erreicht, dass der Farbwert des Pixels auf der Bildebene als Mittelwert der Farben der Schnittpunkte der N Strahlen von der Linse zu diesem Punkt errechnet wird. Da die Strahlen aber aus unterschiedlichen Punkten auf der Linse kommen, laufen sie nach dem Erreichen des Zielpunktes auf der Bildebene in unterschiedliche Richtungen und treffen insbesondere an den Objekträndern auf verschiedene Farben. Das gleiche geschieht auch mit den Objekten, die vor der Bildebene liegen. In diesem Fall wird hier ein Farbwert geliefert, bevor die Strahlen auf die Bildebene treffen. Die Farben für jeden Zielpunkt werden aufaddiert und der Mittelwert gebildet. Ausrechnen des Mittelwertes hat zur Folge, dass der Farbwert des Pixels verfälscht wird. Ein auf diese Art gerendertes Objekt wirkt verschwommen. Ganz anders verhält es sich mit dem Objekt, welches in der Fokusweite und somit auf der Bildebene liegt. Alle Strahlen eines Zielpunktes treffen auf die gleiche Farbe und bilden einen Mittelwert, der eine dem Ursprungspunkt sehr ähnliche Farbe liefert. Da die Farbe an dieser Stelle gleich bleibt ist das Objekt scharf.

Wie schon im vorherigen Kapitel erwähnt, ist die Dicke Linse überflüssig und kann durch eine Dünne Linse ersetzt werden. Die Szenen, die ich bei gleichen Parametern durch verschiedene Linsen laufen ließ, weisen nur bei näherer Betrachtung bei Vergrößerung Unterschiede auf. Insgesamt liefern die Linsen gute Werte, an denen man die Fokussierung gut beobachten kann. Um die Unterschiede gut wahrnehmen zu können, muss bei den Szenen auf das

Vorhandensein der Farbkontraste an den Übergängen zwischen den einzelnen Bestandteilen der Szene geachtet werden. Bei fließenden Übergängen haben zerstreute Pixel oft eine der Umgebung sehr ähnliche Farbe und die Fokussierung lässt sich nur schwer oder überhaupt nicht erkennen. Das gleiche gilt für vor der Bildebene liegende Objekte.

Da die Fokusweite von der Position der Kamera entlang der optischen Achse verläuft und von den Kameradaten abhängig ist, ist das größte Problem bei einer Linsenkamera die Schwierigkeit die Entfernungen der Objekte mit dem bloßen Auge oder aus den Szenendaten abzuschätzen. Die Fokusweite muss durch Probieren für jedes einzelne Objekt in jeder Szene herausgefunden werden. Wenn man die Objekte in der Szene etwas größer beziehungsweise kleine dargestellt haben möchte, ohne die Objekte zu verändern, verschiebt man einfach die Kamera. So können die Szenen auch verschiedenen Positionen aus betrachtet werden.

Literatur :

G. Kolb, D. Mitchell, R. Hanran. A Realistic Camera Model for Computer Graphics

Eugene Hecht. Optik

S. Müller. Script Photorealistische Computergraphik SS 2003

Peter Shirley, R. Keith Morley. Realistic Ray Tracing, S. 68-70