Computergrafik

Vorlesung im Wintersemester 2014/15
Kapitel 2: Ray Tracing
(Teil 2)

Prof. Dr.-Ing. Carsten Dachsbacher
Lehrstuhl für Computergrafik
Karlsruher Institut für Technologie

Ray Tracing

- prinzipielles Vorgehen
  - Erzeugung der Sichtstrahlen durch jeden Pixel (ray generation)
  - Schnittberechnung (ray intersection, ray casting)
    - finde Objekt, das den Sichtstrahl schneidet und dessen Schnittpunkt am nächsten an der Kamera ist
  - Schattierung/Beleuchtungsberechnung (shading)

Materialeigenschaften

- woher kommt das Aussehen von Materialien?

Schattierung/Beleuchtungsberechnung

- klar: Schattierung ist essentiell für dreidimensionalen Eindruck
- Interaktion von
  - Lichtquellen (Intensität, Farbe, Position, …) und
  - Oberflächen (Materialeigenschaften, Orientierung zur LQ)
- zwei „Extrema“ von Reflexion
  - spekulare Reflexion:
    Einfallswinkel = Ausfallswinkel

  - diffuse (Lambertsche) Reflexion:
    gleiche Reflexion in alle Richtungen

Reflexion

- diese Kugeln sehen unterschiedlich aus, weil sie **unterschiedliche Reflexionseigenschaften** haben

Licht-Material-Interaktion

- was passiert, wenn Licht auf eine Oberfläche trifft?
  - ein Teil wird reflektiert
  - ein Teil dringt ins Material ein
  - das Verhältnis ist abhängig vom Einfallswinkel (sog. Fresnel-Effekt) und von den Brechungsindizes beider Materialien

Bilder: Lafortune

Licht-Material Interaktion

- eindringendes Licht wird entweder (sehr schnell) absorbiert: Metalle

- …oder verlässt das Material an anderer Stelle wieder

www.wikipedia.de

- wir befassen uns im Folgenden nur mit der Reflexion an der Oberfläche!

Mehr als nur Oberflächenreflexion

Bilder: Mitsuba

Reflexion

Begriffe und Darstellung

diffus

glänzend, imperfekt spiegelnd
(engl. glossy, specular)

perfekt spiegelnd
(engl. specular)

Reflexion

Begriffe und Darstellung

BRDF

- radiometrisches Konzept um die Reflexion an einem Oberflächenpunkt zu beschreiben: Bidirektionale Reflektanz-Verteilungsfunktion (Bidirectional Reflectance Distribution Function – BRDF)
- Proportionalitätskonstante $f_r(\boldsymbol{\omega_i}, \mathbf{x}, \boldsymbol{\omega_r})$: Verhältnis von
  - Radiance (Strahldichte) die Oberfläche in Richtung $\boldsymbol{\omega_r}$ verlässt, zu
  - einfallender Irradiance (Flußdichte = $\mathrm{W/m^2}$) aus Richtung $\boldsymbol{\omega_i}$
  - **merken Sie sich vorläufig einfach nur:**
    „Verhältnis von ausgehendem zu einfallendem Licht“

BRDF

- BRDFs verwenden ein Referenzkoordinatensystem, in dem die beiden Richtungen durch Winkel ausgedrückt werden:

$$f_r(\boldsymbol{\omega_i}, \mathbf{x}, \boldsymbol{\omega_r}) = f_r(\theta_i, \phi_i, \mathbf{x}, \theta_r, \phi_r)$$

  - Polarwinkel θ $[0..\pi)$ zur Normale, Azimutalwinkel ϕ $[0..2\pi)$
  - die aufspannenden Vektoren heißen: Normale, Tangente, Bitangente

Isotrope und Anisotrope BRDFs

- Isotropie bedeutet Rotationsinvarianz um die Normale
  - $f_r(\theta_i, \phi_i, \mathbf{x}, \theta_r, \phi_r) = f_r(\theta_i, \phi_i + \Delta\phi, \mathbf{x}, \theta_r, \phi_r + \Delta\phi)$
    - Achtung: die Differenz $\phi_i - \phi_r$ ist deswegen nicht irrelevant!
  - isotrope BRDFs sind 3D-Funktionen (unter Weglassen von $\mathbf{x}$)
- Anisotropie: Reflexion abhängig von der Rotation um die Normale
  - anisotrope BRDFs sind 4D-Funktionen (unter Weglassen von $\mathbf{x}$)

Messung von BRDFs – Gonioreflektometer

- nach Murray-Coleman und Smith 1990
  - Radio-/Photometer (bewegt sich relativ zur Materialprobe)
  - Materialprobe (auf Drehteller), Lichtquelle
  - Problem: Verdeckung der LQ durch Detektor
  - mehrere Stunden pro BRDF

Woher kommen BRDFs?

- Messung von realen Materialproben
- Phänomenologisch-motivierte Modelle
  - wenige, intuitiv verständliche Parameter
  - Realismus nur in beschränktem Maße
  - Beispiele
    - Phong [75], Blinn-Phong [77]
    - Ward [92], Lafortune et al. [97], Ashikhmin et al. [00], …
- Physikalisch-motivierte Modelle
  - exaktere Resultate, Einstellung der Parameter schwierig
  - Beispiele
    - Cook-Torrance [81]
    - He et al. [91]

Exkurs: Motivation phänomenologische Modelle

- wie kann man Lichtstrahlen sichtbar machen?

verdünnte Milch:
mehrfache Streuung

fluoreszierende Lösung:
einfache Streuung

Exkurs: Fluoreszenz

- Absorptions- und Emissionsspektrum eines fluoreszierenden Farbstoffs (Eosin Y, Y=„yellowish“)

- ein Streuvorgang ändert die Wellenlänge des Photons: (praktisch) nur einfache Streuung möglich

Exkurs: Gerätschaften

Hullin et al., **Direct Visualization of Real-World Light Transport** *(VMV 2008)*

Exkurs: Oberflächenreflexion

SpiegelInd

Schwarz, spekular

Diffus, weiss

Silber, glossy

Diffus und glossy

Phong-Beleuchtungsmodell

- … ist ein phänomenologisches Modell und modelliert die Reflexion mit drei Komponenten
  - **ambient**: indirekte Beleuchtung, Licht von anderen Oberflächen
  - **diffus**: nach dem Lambertschen Gesetz
  - **spekular**: imperfekte Spiegelung

ambient diffus spekular

- Achtung: wir gehen im Folgenden davon aus, dass alle Richtungsvektoren normiert sind!

Lambertsche (perfekt-diffuse) Reflexion

- effektiv beleuchtete Oberfläche $dA = dA' / \cos\theta$
- Intensität bzw. Energie pro Fläche $I_L / dA \sim \cos\theta$
  - $I_d = k_d \cdot I_\mathrm{L} \cdot \cos\theta = k_d \cdot I_\mathrm{L} \cdot (\mathbf{N} \cdot \mathbf{L})$
  - Intensität der Lichtquelle I_L
  - diffuser Reflexionskoeffizient k_d (wellenlängenabhängig), i.d.R. gegeben für 3 Wellenlängen: rot, grün, blau

Bild: Andrea Weidlich

Lambertsche Reflexion

Bild: Andrea Weidlich

Spekulare Reflexion und Glanzlichter

- durch gerichtete Reflexion entstehen Glanzlichter
- perfekte Spiegelung findet nur in Richtung $\mathbf{R_L}$ statt:

$$\mathbf{R_L} = 2\mathbf{N} \cdot (\mathbf{N} \cdot \mathbf{L}) - \mathbf{L}$$

Bild: Andrea Weidlich

Reflexion

- Spiegelung des Vektors $\mathbf{L}$ an der Oberfläche mit der Normale $\mathbf{N}$:

$$\mathbf{R_L} = 2\mathbf{N} \cdot (\mathbf{N} \cdot \mathbf{L}) - \mathbf{L}$$

Spekulare Reflexion und Glanzlichter

- durch gerichtete Reflexion entstehen Glanzlichter
- perfekte Spiegelung findet nur statt in Richtung $\mathbf{R_L} = \mathbf{2N} \cdot (\mathbf{N} \cdot \mathbf{L}) - \mathbf{L}$
- Stärke der Spiegelung fällt für von $\mathbf{R_L}$ unterschiedl. Richtungen stark ab
  - modelliere Abfall durch $\cos^n \alpha$

- spekulare Reflexion: $I_s = k_s \cdot I_\mathrm{L} \cdot \cos^n \alpha = k_s \cdot I_\mathrm{L} \cdot (\mathbf{R_L} \cdot \mathbf{V})^n$
  - Hinweis: $(\mathbf{R_L} \cdot \mathbf{V}) = (\mathbf{R_V} \cdot \mathbf{L})$
  - spekularer Reflexionskoeffizient k_s
  - „Phong-Exponent“ n

Bild: Andrea Weidlich

Spekulare Reflexion und Glanzlichter

- Auswirkungen des Phong-Exponenten

Bild: Andrea Weidlich

Phong Beleuchtungsmodell

- phänomenologisches Modell, drei Komponenten
  - ambient I_a, diffus I_d und spekular I_s: $I = I_a + I_d + I_s$

$$I = k_a \cdot I_\mathrm{L} + k_d \cdot I_\mathrm{L} \cdot (\mathbf{N} \cdot \mathbf{L}) + k_s \cdot I_\mathrm{L} \cdot (\mathbf{R_L} \cdot \mathbf{V})^n$$

- beachte:
  - k_a, k_d, k_s sind wellenlängenabhängig, ebenso die Lichtintensität I_L
  - diffuse Reflexionen haben meist die Farbe der Oberfläche
  - spekulare Reflexionen
    - haben die Farbe der Oberfläche, wenn es sich um Metalle handelt
    - sonst typischerweise die Farbe der Lichtquelle
  - wir sind nur an Richtungen interessiert, für die die Skalarprodukte positiv sind; oft verwendet man z.B. $(\mathbf{N} \cdot \mathbf{L})^+ := \max\big(0, (\mathbf{N} \cdot \mathbf{L})\big)$

- optional kann das Phong-Beleuchtungsmodell noch durch einen Emissionsterm ergänzt werden

Phong Beleuchtungsmodell

- phänomenologisches Modell, drei Komponenten
  - ambient I_a, diffus I_d und spekular I_s: $I = Ia + I_d + I_s$

$$I = k_a \cdot I_\mathrm{L} + k_d \cdot I_\mathrm{L} \cdot (\mathbf{N} \cdot \mathbf{L}) + k_s \cdot I_\mathrm{L} \cdot (\mathbf{R_L} \cdot \mathbf{V})^n$$

- in der Schreibweise der BRDFs: $f_r(\boldsymbol{\omega_i}, \boldsymbol{\omega_r}) = \frac{\rho_d}{\pi} + \rho_s \cdot \cos^n \phi$
  - $\boldsymbol{\omega_m} = (2\boldsymbol{\omega_i} \cdot \boldsymbol{n})\boldsymbol{n} - \boldsymbol{\omega_i}$ und $\cos\phi = \boldsymbol{\omega_m} \cdot \boldsymbol{\omega_r}$
  - $\boldsymbol{\omega_m}$ ist hier die Spiegelungsrichtung von $\boldsymbol{\omega_i}$ an $\boldsymbol{n}$, $\boldsymbol{\omega_r}$ die Richtung für die wir die BRDF auswerten!
  - wo ist der Kosinusterm im diffusen Anteil? (siehe Reflexionsintegral, Distributed Raytracing)

Phong Beleuchtungsmodell

Bild: Andrea Weidlich

Ray Tracing, Stereo-Rendering

- Grundidee: für jedes Auge ein Bild

Ray Tracing Pseudocode

```
for ( y = 0; y < height; y++ ) {
  for ( x = 0; x < width; x++ ) {
    u = l + (r-l) * (x+0.5) / width;
    v = t + (b-t) * (y+0.5) / height;
    s = ...;
    d = normalize( s );

    // finde nächsten Schnittpunkt
    intersection = NULL;
    float t = FLOAT_MAX;

    for ( each object ) {
      t‘ = intersect( object, e, d );
      if ( t‘ > 0 && t‘ < t ) {
        intersection = object;
        t = t‘;
      }
    }

    // Beleuchtungsberechnung
    if ( intersection != NULL ) { ... }
} }
```

Beleuchtungsberechnung im Ray Tracer

```
struct vec3 {          // oder awir verwenden in der Übung die
  float x, y, z;       // OpenGL Mathematics (GLM) Library
  ...
};

class Material {
  vec3  ka, kd, ks;    // Reflexionsparameter (schöner: Spektrum)
  float n;             // Phong-Exponent
  ...
};

struct Intersection {
  vec3 p, n, v;        // Position, Normale,
                       // Vektor zur Kamera/zum Startpunkt des Strahls
  Material *material; // Reflexionsparameter
};

class PointLight : public LightSource {
  vec3 pos,            // Position
       I_l;            // Intensität der LQ (schöner: Spektrum)
  ...
};
```

Beleuchtungsberechnung im Ray Tracer

```
vec3 PointLight::computeDirectLight( const Intersection &i, ... )
{
  // ambienter Term
  vec3 I = i.material->ka * I_l;

  vec3 L = normalize( pos - i.p );
  float  NdotL = dot( i.n, L );

  if ( NdotL > 0 ) {
    // diffuser Term
    I += i.material->kd * I_l * NdotL;

    // spekularer Term
    vec3  R = 2.0f * i.n * NdotL – L;
    float RdotV = dot( R, V );
    if ( RdotV > 0 ) {
      I += i.material->ks * I_l * powf( RdotV, i.material->n );
    }
  }
};
```

$$I = k_a \cdot I_\mathrm{L} + k_d \cdot I_\mathrm{L} \cdot (\mathbf{N} \cdot \mathbf{L}) + k_s \cdot I_\mathrm{L} \cdot (\mathbf{R_L} \cdot \mathbf{V})^n$$

Beleuchtungsberechnung im Ray Tracer

```
vec3 PointLight::computeDirectLight( const Intersection &i, ... )
{
  // ambienter Term
  vec3 I = i.material->ka * I_l;

  vec3 L = normalize( pos - i.p );
  float  NdotL = dot( i.n, L );
```

$$I = k_a \cdot I_\mathrm{L} + k_d \cdot I_\mathrm{L} \cdot (\mathbf{N} \cdot \mathbf{L}) + k_s \cdot I_\mathrm{L} \cdot (\mathbf{R_L} \cdot \mathbf{V})^n$$

die Intensität einer „realistischen“ Punktlichtquelle würde mit dem Abstandsquadrat abfallen:

```
float dist = length( pos – i.p  );
...
I += i.material->kd * I_l * NdotL /
     ( dist * dist );
```

```
    if ( RdotV > 0 ) {
      I += i.material->ks * I_l * powf( RdotV, i.material->n );
    }
  }
};
```

Schattierung von Dreiecksnetzen

Beleuchtungsberechnung mit welcher Normale?

- Dreiecke: stückweise lineare Approximation einer (möglicherweise gekrümmten) Oberfläche
- **Flat Shading**: Verwendung der Dreiecksnormale für Beleuchtung

Schattierung von Dreiecksnetzen

Beleuchtungsberechnung mit welcher Normale?

- Illusion einer glatten, gekrümmten Fläche durch
  - Normalenvektoren an den Vertizes/Eckpunkten („Vertex-Normalen“)
  - Interpolation der Normalen

- Vertex-Normalen können entweder
  - berechnet werden (z.B. bei der Triangulierung parametrischer Flächen)
  - direkt aus einem Dreiecksnetz bestimmt werden

Schattierung von Dreiecksnetzen

Normalen für Dreiecksnetze

- Vertex-Normalen bestimmen über
  - gewichtete Mittelung der Normalen angrenzender Flächen
  - Normalen für ein Dreieck über Kreuzprodukt

- Vorgehen
  - berechne Normale für jedes Dreieck
  - für jeden Vertex: summiere die Normalen aller angrenzenden Dreiecke
    - optional: skaliere Normalen mit der Dreiecksfläche (ohne Zutun im Kreuzprodukt)
  - **normalisiere** die Vertex-Normalen
    - NICHT durch die Anzahl der Normalen teilen!

Schattierung von Dreiecksnetzen

Normalen für Dreiecksnetze

- scharfe Kanten brauchen mehrere Normalen
  - Kante, wenn der Winkel zweier benachbarter Dreiecke zu groß ist
- bei Dreiecksnetzen werden pro Dreieck typischerweise 3 Eckpunkte, 3 Normalen (und zusätzliche Attribute, z.B. Farben) gespeichert

glatte Schattierung

scharfe Kanten

Schattierung von Dreiecksnetzen

Interpolation von Normalen

- lineare komponentenweise Interpolation der Vertex-Normalen anhand der baryzentrischen Koordinaten

- Beispiel:
  - geg.: Normalen $\mathbf{n_1}, \mathbf{n_2}, \mathbf{n_3}$ eines Dreiecks $\triangle\,(\mathrm{P}_1, \mathrm{P}_2, \mathrm{P}_3)$
  - ges.: interpolierte Normale $\mathbf{n_Q}$ an einem Punkt Q auf dem Dreieck
  - Lösung: berechne $\lambda_1, \lambda_2, \lambda_3$, dann ist $\mathbf{n_Q} = \lambda_1 \mathbf{n_1} + \lambda_2 \mathbf{n_2} + \lambda_3 \mathbf{n_3}$

- Beleuchtungsberechnung mit der Normale $\mathbf{n_q}/|\mathbf{n_q}|$
  - lineare komponentenweise Interpolation ist nicht längenerhaltend!

- Beleuchtungsberechnung mit interpolierten Normalen nennt man **Phong Shading** (≠ Phong Beleuchtungsmodell)

Schattierung von Dreiecksnetzen

- Flat Shading (links) und Phong Shading (rechts)

Licht und Schatten

- bisher: eine Oberfläche wird immer beleuchtet
- aber: andere Objekte der Szene können Schatten werfen
- Lösung: sende einen „Schattenstrahl“ zur Lichtquelle hin
  - teste, ob dieser Strahl auf dem Weg **von der Oberfläche zur Lichtquelle** ein anderes Objekt schneidet

Die Beiträge mehrerer Lichtquellen werden einfach aufsummiert (Superpositionsprinzip)

Schattenberechnung im Ray Tracer

```
vec3 PointLight::computeDirectLight( const Intersection &i, ... ) {
  // ambienter Term
  vec3 I = i.material->ka * I_l;

  vec3 L = pos - i.p;

  // Schattenstrahl
  float dist2Light = length( L );

  for ( each object ) {
    t‘ = intersect( object, i.p, L );
    if ( t‘ > 0 && t‘ < dist2Light )
      return I;
  }
  ...
};
```

- i.d.R. zwei unterschiedliche Routinen in einem Ray Tracer
  - teste ob irgendein Schnittpunkt für $t \in (0; t_{max})$
  - suche den nächsten Schnittpunkt für $t > 0$

Schattenberechnung im Ray Tracer

- Problem: endliche Genauigkeit bei der Gleitkommadarstellung
  - wird der Schattentest so durchgeführt, wie auf der vorherigen Folie beschreiben, treten Artefakte auf:
    der Strahl schneidet u.U. die Oberfläche von der er startet nochmal!

- prinzipiell zwei Möglichkeiten
  - teste explizit, ob nochmals dasselbe Objekt geschnitten wurde:
    funktioniert nur bei konvexen Objekten
  - starte etwas weiter weg von der Oberfläche
    - `t‘ = intersect( object, i.p + epsilon * i.n, L );`

$\epsilon = 0$

$\epsilon = 10^{-7}$

Ray Tracing Pseudocode

```
class Ray {
  vec3   origin, direction; // Startpunkt und Richtung des Strahls
  float  t;                 // Strahlparameter
  void * object;            // Zeiger auf evtl. getroffenes Objekt
  ...
};

for ( y = 0; y < height; y++ ) {
  for ( x = 0; x < width; x++ ) {
    u = l + (r-l) * (x+0.5) / width;
    v = t + (b-t) * (y+0.5) / height;
    s = ...;
    d = normalize( s );

    // finde nächsten Schnittpunkt
    intersection = NULL;
    float t = FLOAT_MAX;

    for ( each object ) {
      t‘ = intersect( object, e, d );
      if ( t‘ > 0 && t‘ < t ) {
        intersection = object;
        t = t‘;
    } }

    // Beleuchtungsberechnung
    if ( intersection != NULL ) {
      computeDirectLight ...
} } }
```

Strahlerzeugung (abh. vom Kameramodell)

```
generateRay( Ray &ray, int x, int y );
```

Strahlverfolgung und Beleuchtung

```
vec3 color =
  raytrace( Ray *ray, ... );
```

Ray Tracing Pseudocode

```
class Ray {
  vec3   origin, direction; // Startpunkt und Richtung des Strahls
  float  t;                 // Strahlparameter
  void * object;            // Zeiger auf evtl. getroffenes Objekt
  ...
};

for ( y = 0; y < height; y++ ) {
  for ( x = 0; x < width; x++ ) {
    u = l + (r-l) * (x+0.5) / width;
    v = t + (b-t) * (y+0.5) / height;
    s = ...;
    d = normalize( s );

    // finde nächsten Schnittpunkt
    intersection = NULL;
    float t = FLOAT_MAX;

    for ( each object ) {
      t' = intersect( object, e, d );
      if ( t' > 0 && t' < t ) {
        intersection = object;
        t = t';
    } }

    // Beleuchtungsberechnung
    if ( intersection != NULL ) {
      computeDirectLight ...
} } }
```

Strahlerzeugung (abh. vom Kameramodell)
`generateRay( Ray &ray, int x, int y );`

Schnittpunktberechnung
`bool cast( Ray *ray, float maxDist );`

Arten von Lichtquellen

- Punktlichtquelle
  - definiert durch Position $\mathbf{p}$ und Intensität I_L (W/sr)
  - reale Lichtquellen: Abfall der Intensität mit dem Abstandsquadrat
- paralleles Licht/direktionale Lichtquelle
  - definiert durch Richtung $\mathbf{d}$ und Flussdichte E (W/m^2)
  - Bsp. Sonnenlicht ist annähernd parallel
- Andere Lichtquellentypen
  - Strahler (Spot-Lights): Lichtkegel, Abstrahlungscharakteristik mit $\cos^n \theta$
  - Flächenlichtquellen
  - Punktlichtquellen mit Richtungscharakteristik

Primär- und Sekundärstrahlen

- Ray Tracing basiert auf den Gesetzen der geometrischen Optik
- Grundidee: Strahlverfolgung vom Auge aus
- Rekursion um Spiegelung und Lichtbrechung darzustellen
  → Sekundärstrahlen (**R**, **T**)

P: Primärstrahl

S: Schattenstrahl

R: Reflexion

T: Transmission

Spiegelungen, Reflexionsstrahlen

- Spiegelungen der Szene auf reflektierenden Oberflächen
  - Sichtstrahl $\mathbf{V}$ wird an der Oberfläche reflektiert → Reflexionsstrahl $\mathbf{R}$
- behandle Reflexionsstrahl (fast so) wie einen Sichtstrahl: berechne Schnittpunkt und gebe Farbe zurück
  - Beleuchtungsberechnung für die „Blickrichtung“ $\mathbf{R}$
  - rekursive Verfolgung von (Mehrfach-)Reflexionen
  - weitere Schattenstrahlen
- addiere Farbe des Reflexionsstrahl zur Farbe am Ausgangspunkt
  - weiterer Koeffizient k_r (oder k_t)

Ray Tracing Pseudocode

```
class Ray {
  vec3   origin, direction; // Startpunkt und Richtung des Strahls
  float  t;                 // Strahlparameter
  void * object;            // Zeiger auf evtl. getroffenes Objekt
  ...
};

for ( y = 0; y < height; y++ ) {
  for ( x = 0; x < width; x++ ) {
    u = l + (r-l) * (x+0.5) / width;
    v = t + (b-t) * (y+0.5) / height;
    s = ...;
    d = normalize( s );

    // finde nächsten Schnittpunkt
    intersection = NULL;
    float t = FLOAT_MAX;

    for ( each object ) {
      t‘ = intersect( object, e, d );
      if ( t‘ > 0 && t‘ < t ) {
        intersection = object;
        t = t‘;
    } }

    // Beleuchtungsberechnung
    if ( intersection != NULL ) {
      computeDirectLight ...
} } }
```

Strahlverfolgung und Beleuchtung

```
vec3 color =
  raytrace( Ray *ray, ... );
```

um rekursive Strahlen zu verfolgen

Ray Tracing Pseudocode

```
vec3 raytrace( Ray *ray, ... ) {
  // Farbe
  vec3 color = 0.0f;

  if ( !cast( ray, FLOAT_MAX ) )
    return color;

  for ( jede Lichtquelle )
    color += computeDirectLight( ... );

  if ( Fläche ist spiegelnd ) {
    // berechne Reflexionsstrahl
    Ray reflect = ...;
    color += i.kr * raytrace( reflect, ... );
  }

  return color;
}
```

Spiegelungen, Reflexionsstrahlen

Wie ist die Rekursionstiefe?

- vorgegebene, maximale Anzahl von Reflexionen
- Rekursion bis der Beitrag zur Farbe vernachlässigbar wird
  - Beitrag der 1. Reflexion: $k_{r,1}$
  - Beitrag der 2. Reflexion: $k_{r,1} \cdot k_{r,2}$
  - verfolge bis $\prod_i k_{r,i} < \epsilon_r$
  - Achtung: der tatsächliche Beitrag hängt auch von Intensität der LQ ab

Metallkugeln – Rekursionstiefe 0

- Bild: Pat Hanrahan

Metallkugeln – Rekursionstiefe 1

- Bild: Pat Hanrahan

Metallkugeln – Rekursionstiefe 2

- Bild: Pat Hanrahan

Rekursionstiefe 5

- Bild: Pat Hanrahan

Glaskugeln – Rekursionstiefe 0

- Bild: Pat Hanrahan

Glaskugeln – Rekursionstiefe 1

- Bild: Pat Hanrahan

Glaskugeln – Rekursionstiefe 2

- Bild: Pat Hanrahan

Glaskugeln – Rekursionstiefe 5

- Bild: Pat Hanrahan

Snellsches Brechungsgesetz

- … beschreibt die Richtungsänderung einer Welle (Licht) beim Übergang von einem Medium in eines mit anderer Brechzahl

- Licht bewegt sich in unterschiedlichen Medien unterschiedlich schnell
  - (reelle) Brechzahl $\eta = {}^{c_0}/_{c_\eta}$
  - c_0 Phasengeschwindigkeit im Vakuum
  - c_η Phasengeschwindigkeit im Medium

- Brechzahl ist i.d.R. wellenlängenabhängig $\eta(\lambda)$
  - wellenlängenabhängige Ausbreitungsgeschwindigkeit von Wellen (Dispersion) → Aufspaltung in unterschiedliche Wellenlängen

Bilder: Wikipedia u.a.

Snellsches Brechungsgesetz

- Brechungsgesetz: $\eta_i \sin\theta_i = \eta_t \sin\theta_t$
  - Brechung bei Übergang ins optisch dichtere Medium ($\eta_t > \eta_i$) zum Lot hin
  - beim Übergang ins optisch dünnere Medium vom Lot weg
  - Fresnel-Effekt: Verteilung der Strahldichte
- Grenzwinkel der Totalreflexion
  - Licht wird an der Grenzfläche von dichterem zu dünnerem Medium vollständig reflektiert: $\sin\theta_c = \eta_i/\eta_t$

- Transmissionsvektor $\mathbf{T}$ (Herleitung siehe Fund. of Computer Graphics)

$$\mathbf{T} = -\frac{\sin\theta_t}{\sin\theta_i}(\mathbf{I} - \mathbf{N}\cos\theta_i) - \mathbf{N}\cos\theta_t$$

$$= -\frac{\eta_i}{\eta_t}\mathbf{I} + \left(\frac{\eta_i}{\eta_t}\cos\theta_i - \sqrt{1 - \left(\frac{\eta_i}{\eta_t}\right)^2 (1 - \cos^2\theta_i)}\right)\mathbf{N}$$

- $\mathbf{I}$, $\mathbf{N}$, $\mathbf{R}$ und $\mathbf{T}$ liegen in einer Ebene

Fresnel-Effekt, Lichtbrechung und Reflexion

Bilder: http://www.fizkapu.hu/fizfoto/fizfoto6.html

Totalreflexion

Bilder: http://www.fizkapu.hu/fizfoto/fizfoto6.html

Refraktion: Snell‘s Window, Optical Manhole

Fig. 3.7A The optical manhole. From under water, the entire celestial hemisphere is compressed into a circle only 97.2° across. The dark boundary defining the edges of the manhole is not sharp due to surface waves. The rays are analogous to the crepuscular type seen in hazy air, Section 1.9. (Photo by D. Granger)

Fig. 3.7B The optical manhole. Light from the horizon (angle of incidence = 90°) is refracted downward at an angle of 48.6°. This compresses the sky into a circle with a diameter of 97.2° instead of its usual 180°.

aus “Color and Light in Nature”, Lynch and Livingstone

Ray Tracing Pseudocode

```
vec3 raytrace( Ray *ray, ... ) {
  vec3 color = 0.0f;

  if ( !cast( ray, FLOAT_MAX ) )
    return color;

  for ( jede Lichtquelle )
    color += computeDirectLight( ... );

  if ( Fläche ist spiegelnd ) {
    // berechne Reflexionsstrahl
    Ray reflect = ...;
    color += i.kr * raytrace( reflect, ... );
  }

  if ( Fläche ist transparent ) {
    // berechne Transmissionsstrahl
    Ray refract = ...;
    if ( Totalreflexion )
      color += i.kr * raytrace( reflect, ... );
    else
      color += i.kt * raytrace( refract, ... );
  }

  return color;
}
```

Diese rekursive Formulierung heißt Whitted-Style Ray Tracing (nach Turner Whitted)

Recap Ray Tracing

- verfolge den Weg des Lichts – das den Betrachter erreicht – zurück
- Lichttransport basiert auf geometrischer Optik
- bildbasiertes Verfahren (Image-Order Rendering)

- Ablauf
  - Erzeugung von Primärstrahlen = Kamera-/Sichtstrahlen
  - finde Schnittpunkt mit nächster Oberfläche
  - Schattierung/Beleuchtungsberechnung
    - **lokale** Beleuchtungsberechnung und **globale** Schattenstrahlen
    - rekursive Berechnung von Spiegelung und Refraktion

Ray Tracing und Aliasing

- Aliasing-Effekte entstehen durch (falsche) Abtastung von Signalen
- auch das Bild einer 3D-Szene ist ein (zweidimensionales) Signal
  - Farbe eines Pixels bestimmt durch Strahl durch seinen Mittelpunkt
  - d.h. an den Pixelmittelpunkten tasten wir das Bildsignal ab
  - das Bildsignal ist nicht bandbegrenzt (gilt v.a. für die Objektkanten)
  - durch quadratische Form der Pixel sehen wir zusätzlich Treppenstufen

Ray Tracing und Aliasing

- Rasterbilder (recap)
  - Vorteil bei der Darstellung des Bilder: wiederholter Bildaufbau ist unabhängig von der Komplexität der Szene (Frame Buffer)
- Eigenschaften
  - eine uniforme, diskrete Abtastung eines Signals
  - eine stückweise-konstante Funktion (hier: Illustration in 1D)
  - Aliasing, Moiré-Effekte, „Jaggies“ (Treppenstufen-Effekte)

Bild: Wikipedia

Bild, Signal und Abtastung

- Graustufenbild einer Funktion $f(d) = \cos ad^2$ mit $d \in [-1,1]^2$, $a \in \mathbb{R}$

```
#define WIDTH  128
#define HEIGHT 128

unsigned char img[ WIDTH * HEIGHT ];

for ( y = 0; y < HEIGHT; y++ ) {
  for ( x = 0; x < WIDTH; x++ ) {
    float dx = (float)( 2*x - WIDTH  ) / (float)WIDTH;
    float dy = (float)( 2*y - HEIGHT ) / (float)HEIGHT;
    float d2 = dx * dx + dy * dy;

    // f(d) mit a = 64
    float f = cosf( 64.0f * d2 );

    // abbilden von f(d) auf [0..255]
    img[ x + y * WIDTH ] =
        ( f + 1.0f ) * 127.5f;
}  }
```

Bild, Signal und Abtastung

- Graustufenbild einer Funktion $f(d) = \cos ad^2$ mit $d \in [-1,1]^2$, $a \in \mathbb{R}$

WIDTH=HEIGHT=128

WIDTH=HEIGHT=64

WIDTH=HEIGHT=32

Bild, Signal und Abtastung

- Nyquist-Shannon-Abtasttheorem:
  ein kontinuierliches, bandbegrenztes Signal mit einer max. Frequenz f_{max} muss mit einer Frequenz größer als $2f_{max}$ abgetastet werden, damit aus dem diskreten Signal das Ursprungssignal exakt rekonstruiert werden kann
- Beispiel: $f_{abtast} > 2f_{max}$ (letzte Zeile $f_{abtast} = 2f_{max}$, nicht eindeutig)

Bild, Signal und Abtastung

- Nyquist-Shannon-Abtasttheorem:
  ein kontinuierliches, bandbegrenztes Signal mit einer max. Frequenz f_{max} muss mit einer Frequenz größer als $2f_{max}$ abgetastet werden, damit aus dem diskreten Signal das Ursprungssignal exakt rekonstruiert werden kann
- Beispiel: $f_{abtast} \leq 2f_{max}$ —— orig. Signal ------- rekonstr. Signal

Bild, Signal und Abtastung

- Aliasing-Effekt: Fehler beim Abtasten von Signalen
  - im rekonstruierten Signal (z.B. Darstellung des Bildes am Monitor) treten Frequenzen auf, die im Original nicht enthalten sind

- 1D- und 2D-Version unseres Beispiels: Original und Rekonstruktion

Unterschiedliche Rekonstruktionsfilter

WIDTH=HEIGHT=128

WIDTH=HEIGHT=64

WIDTH=HEIGHT=32

Bild, Signal und Abtastung

- Aliasing-Effekt: Fehler beim Abtasten von Signalen
  - im rekonstruierten Signal (z.B. Darstellung des Bildes am Monitor) treten Frequenzen auf, die im Original nicht enthalten sind
- Lösungsansätze
  - Filterung des Signals vor der Abtastung (hohe Frequenzen „entfernen“): im allgemeinen Fall nicht möglich
  - höhere Abtastung des Signals und anschließend Mitteln (rechtes Bild)

Abtastung ausreichend

Abtastung zu niedrig → Aliasing

Überabtastung

Bsp. Aliasing

- …das klassische Beispiel für Aliasing in der Computergrafik

Ray Tracing und Aliasing

Anti-Aliasing Strategien

- Anti-Aliasing: Versuch Aliasing zu reduzieren oder zu vermeiden
- wir können das Bildsignal i.A. nicht bandbegrenzen
- es bleibt i.d.R. nur Überabtastung (Supersampling)
  - im Fall des Aliasing durch Objektkanten:
    ein Objekt bedeckt nur einen Teil eines Pixels – wir versuchen durch Überabtasten (= Supersampling) herauszufinden, welchen Beitrag die Objekte zu den Pixelfarben liefern

Anti-Aliasing Strategien

Uniformes Supersampling

- statt Abtastung eines Punktes innerhalb eines Pixels (ein Sample) tasten wir k^2-mal in äquidistanten Intervallen ab

1 Pixel – 1 Sample

1 Pixel – mehrere Samples

- nehme den Durchschnitt der k^2-Farben als Pixel-Farbe
- Primärstrahlen dann auch durch nicht-ganzzahlige Pixel-Positionen auf der Bildebene

Anti-Aliasing Strategien

Uniformes Supersampling

- Abtastung des Pixels

1 Sample pro Pixel

4 Samples pro Pixel

9 Samples pro Pixel

Anti-Aliasing Strategien

Uniformes Supersampling

- Abstand zwischen den Samples im Bild muss immer gleich sein

- so darf es nicht sein:

mehrere Samples an derselben Stelle

Anti-Aliasing Strategien

Adaptives Supersampling (heute kaum noch verwendet)

- ist die (Farb-)Differenz benachbarter Samples zu groß, dann unterteile den Pixel in 4 Bereiche und verfolge weitere Strahlen
- „einfarbige“ Pixel benötigen im Schnitt 1 Sample
- Kanten werden trotzdem geglättet

- ursprüngliche Samples
- 1. Unterteilung
- 2. Unterteilung
- 3. Unterteilung

Anti-Aliasing Strategien

Stochastisches Sampling

- uniformes Supersampling reicht oft nicht, um hohe Frequenzen aufzulösen → stochastisch = verwende zufällige Samples in einem Pixel
- noch besser: zufällige Samples mit Stratifikation (Stratified Sampling)
  - unterteile Pixel in ein Gitter, wähle einen Zufallspunkt pro Zelle
  - sehr oft verwendeter Ansatz
- nicht-uniformes Abtasten: reduziert Aliasing, aber führt zu mehr Rauschen (von unserer Wahrnehmung als weniger störend empfunden)

Ray Tracing in PostScript?

- PS ist eine Stacksprache (wie z.B. auch FORTH)

```
%!OPS-1.0 %%Creator: HAYAKAWA,Takashi<h-takasi@isea.is.titech.ac.jp>
/A/copy/p/floor/q/gt/S/add/n/exch/i/index/J/ifelse/r/roll/w/div/H{{loop}stopped
Y}def/t/and/C/neg/T/dup/h/exp/Y/pop/d/mul/s/cvi/e/sqrt/R/rlineto{load def}H 300
T translate(V2L&1i2A00053r45hNvQXz&vUX&UOvQXzFJ!FJ!J!O&Y43d9rE3IaN96r63rvx2dcaN
G&140N7!U&4C577d7!z&&93r6IQO2Z4o3AQYaNlxS2w!!f&nY9wn7wpSps1t1S!D&cjS5o32rS4oS3o
Z&blxC1SdC9n5dh!I&3STinTinTinY!B&V0R0VRVC0R!N&3A3Axe1nwc!I&993dC99Cc96raN!a&1CD
E&YYY!F&&vGYx4oGbxSd0nq&3IGbxSGY4Ixwca3AlvvUkbQkdbGYx4ofwnw!&vlx2w13wSb8Z4wS!J!
c&j1idj2id42rd!X&4I3Ax52r8Ia3A3Ax65rTdCS4iw5o5IxnwTTd32rCST0q&eCST0q&D1!&EYE0!J
&EYEY0!J0q!x&jd5o32rd4odSS!K&WCVW!Q&31C85d4!k&X&E9!&1!J!v&6A!b&7o!o&1r!j&43r!W)
{( )T 0 4 3 r put T(/)q{T(9)q{cvn}{s}J}{($)q{[}{]}J}J cvx}forall 270{def}H
K{K{L setgray moveto B fill}for Y}for showpage
```

Distributed Ray Tracing

- „Probleme“ des Whitted-Style Ray Tracing Verfahrens
  - die Bilder sehen zu makellos aus:
    - perfekte Spiegelung und Transmission
    - harte Schattenkanten
    - unendliche Schärfentiefe etc.

- Beispiel: Schattenstrahlen

Weiche Schatten

- reale Lichtquellen besitzen endliche Ausdehnung

Bewegungsunschärfe

- Verteilung der Strahlen in der Zeit
  - Objekte bewegen sich in einem Zeitintervall $[t; t + \Delta t)$
  - erzeuge für jeden Pixel n Strahlen für einen Zeitpunkt $t' \in [t; t + \Delta t)$
    - führe Ray Tracing mit der Szene zum Zeitpunkt t' durch
    - mittle Farbwerte
- vgl. Belichtungszeit bei realen Kameras

Tiefenunschärfe

Modell der „dünnen Linse“

- Abtasten der Linsenfläche
  - wähle Punkt auf der Bildebene $\mathbf{P_b}$ und der Linse $\mathbf{P_l}$
  - berechne daraus Punkt auf der Fokusebene $\mathbf{P_f}$
  - schicke Strahl von $\mathbf{P_l}$ durch $\mathbf{P_f}$

Tiefenunschärfe

[Cook et al. 1984]

Rob Cook is among the first people to have been awarded an Oscar for software. He is the co-architect and primary author of Pixar's RenderMan.

Und was ist mit …

Und was ist mit …

Und was ist mit …

Und was ist mit …

Und was ist mit …

Imperfekte Spiegelung und Transmission

Distributed Ray Tracing

- mehrere Strahlen für Spiegelung und Transmission
  - an jedem Schnittpunkt werden viele Sekundärstrahlen weiterverfolgt
  - (quasi-)zufällige Richtung
  - Gewichtung nach der BRDF

- wie viele Strahlen werden benötigt?
  - durch Rekursion verzweigt der „Strahlenbaums“ schon beim Whitted-Style Ray Tracing
  - Explosion bei mehreren Sekundärstrahlen an jedem Punkt

Imperfekte Spiegelung und Transmission

- je 1 zufälliger Sekundärstrahl: Zufall → Bildrauschen

Imperfekte Spiegelung und Transmission

- je 16 zufällige Sekundärstrahlen

Imperfekte Spiegelung und Transmission

- je 256 zufällige Sekundärstrahlen