Bericht zur 04 Schnittberechnung II

Marcus Baetz, Andreas Kiauka, Robert Dziuba

13. Dezember 2015

Inhaltsverzeichnis

1 Aufgabenstellung

1.1 Erweiterung an den Light-Klassen

Die Implementierung der Beleuchtung in unserem Raytracer sollte erweitert werden. Genauer gesagt sollte der Raytracer um die Darstellung von Schatten und Reflexionen erweitert werden.

1.1.1 castShadows

Die Oberklasse Light sollte um das Attribut castShadows vom Typ Boolean erweitert werden.

1.1.2 illuminates(...)

In der der Methode illuminates(...) sollte nun ermittelt werden, ob ein Objekt zwischen der Lichtquelle und und dem Punkt auf der Oberfläche ist.

1.2 Erweiterung an den Material-Klassen

Auch die Materialklassen sollten erweitert werden.

1.2.1 colorFor(...)

Die Methode colorFor erhielt als zusätzlichen Parameter einen Tracer. Dies ist ein Objekt, welches eine Funktion zum Raytracen zur Verfügung stellt.

1.2.2 ReflectiveMaterial

Bei diesem Matrial handelt es sich um ein Material für einen perfekt diffus reflektierenden Körper mit Glanzpunkt und Reflektion.

2 Lösungstrategien

Die Aufgabe wurde von der Gruppe auf verschiedenen Branches erarbeitet. Am Ende wurde alles auf den Master-Branch gemerged.

3 Implementierung

3.1 illuminates(...)

Die Methode illuminates der Klasse SpotLight nach der Erweiterung.

```
1  public boolean illuminates(final Point3 point, final World
       world) {
2    if (point == null) {
3      throw new IllegalArgumentException("The point cannot be
           null!");
4    }
5    if (world == null) {
6      throw new IllegalArgumentException("The world cannot be
           null!");
7    }
8    if(Math.acos(direction.dot(directionFrom(point).mul(-1)))
         <= halfAngle) {
9      if(castsShadow) {
10       final Ray r = new Ray(point, directionFrom(point));
11       final double tl = r.tOf(position);
12       for (final Geometry g : world.geometries) {
13         final Hit h = g.hit(r);
14         if ((h != null && h.t >= 0.0001 && h.t < tl)) {
15           return false;
16         }
17       }
18     }
19
20   return true;
21   }
22   return false;
23 }
```

Die Methode illuminates(...) der Klasse PointLight.

```
1  public boolean illuminates(final Point3 point,final World
       world) {
2    if (point == null) {
3      throw new IllegalArgumentException("The point cannot be
           null!");
4    }
5    if (world == null) {
6      throw new IllegalArgumentException("The world cannot be
           null!");
7    }
8
9    if(castsShadow) {
10     final Ray r = new Ray(point, directionFrom(point));
11     final double tl = r.tOf(position);
12     for (final Geometry g : world.geometries) {
13       final Hit h = g.hit(r);
14       if ((h != null && h.t >= 0.0001 && h.t < tl)) {
15         return false;
16       }
17     }
18   }
19   return true;
20 }
```

3.2 colorFor(...) ReflectiveMaterial

Die Methode colorFor der Klasse reflective Material.

```
1  public Color colorFor(Hit hit, World world, Tracer tracer)
       {
2    if (hit == null) {
3      throw new IllegalArgumentException("The hit cannot be
         null!");
4    }
5    if (world == null) {
6      throw new IllegalArgumentException("The world cannot be
         null!");
7    }
8    Color basicColor = new Color(0, 0, 0);
9    final Vector3 e = hit.ray.d.mul(-1).normalized();
10   final Point3 h = hit.ray.at(hit.t);
11   final Ray refRay = new Ray(h,hit.ray.d.normalized().mul
       (-1).reflectedOn(hit.n));
12   for (Light light : world.lights) {
13     Vector3 l = light.directionFrom(h);
14     Vector3 rl = l.reflectedOn(hit.n);
15     if (light.illuminates(h, world)) {
16       basicColor = basicColor.add(
17         light.color.mul(diffuse)
18           .mul(Math.max(0, hit.n.dot(l.normalized())))
19           )
20       ).add(
21         specular
22           .mul(light.color)
23           .mul(Math.pow(
24             Math.max(0, rl.dot(e)), exponent)
25           )
26       );
27     }
28     basicColor = basicColor.add(reflection.mul(tracer.
         reflection(refRay,world)));
29   }
30   return diffuse.mul(world.ambientLight).add(basicColor);
31 }
```

3.3 colorFor() TransparenMaterial

```
1  /**
2  * Returns the right illuminated color for the hit point
3  *
4  * @param hit    The Hit-Object for the hit Point
5  * @param world  The WorldObject for this scene
6  * @param tracer calculates the color
7  * @return the Color-Object for the hit point
8  */
9  @Override
10 public Color colorFor(Hit hit, World world, Tracer tracer)
       {
11 if (hit == null) {
12 throw new IllegalArgumentException("The hit cannot be null!
       ");
13 }
14 if (world == null) {
15 throw new IllegalArgumentException("The world cannot be
       null!");
16 }
17
18 Color basicColor = new Color(0, 0, 0);
19
20 double ref;
21 Normal3 n;
22 final Vector3 e = hit.ray.d.mul(-1).reflectedOn(hit.n);
23 if(e.dot(hit.n) < 0){
24 ref = iOR/ImageSaver.raytracer.iOR;
25 n = hit.n.mul(-1);
26
27 }else{
28 ref = ImageSaver.raytracer.iOR/iOR;
29 n = hit.n;
30 }
31 double cosA1 = n.dot(e);
32 double co = 1 - (ref * ref) * (1 - cosA1 * cosA1);
33 double a=1;
34 if(co>=0) {
35 double cosA2 = Math.sqrt(co);
36 final double a0 = Math.pow((ImageSaver.raytracer.iOR - iOR)
       / (ImageSaver.raytracer.iOR + iOR), 2);
37 a = a0 + (1 - a0) * Math.pow(1 - cosA1, 5);
```

```
38 final double b = 1 - a;
39 Vector3 t = hit.ray.d.mul(ref).sub(n.mul(cosA2-ref*cosA1));
40 Ray refractionRay = new Ray(hit.ray.at(hit.t+0.00001),t);
41 Tracer tracer2 = new Tracer(tracer.recursionDepth);
42 basicColor = basicColor.add(tracer2.reflection(
       refractionRay,world).mul(b));
43 }
44 Vector3 r =  hit.ray.d.normalized().add(n.mul(2*cosA1));
45 final Point3 p = hit.ray.at(hit.t-0.00001);
46 Ray reflectionRay = new Ray(p,r);
47
48 for (Light light : world.lights) {
49
50 Vector3 l = light.directionFrom(p);
51 Vector3 rl = l.reflectedOn(hit.n);
52 if (light.illuminates(p, world)) {
53 basicColor = basicColor.add(
54 light.color.mul(diffuse)
55 .mul(Math.max(0, hit.n.dot(l.normalized())))
56 )
57 ).add(
58 specular
59 .mul(light.color)
60 .mul(Math.pow(
61 Math.max(0, rl.dot(e)), exponent)
62 )
63 );
64 }
65 basicColor = basicColor.add(reflection.mul(tracer.
       reflection(reflectionRay,world)).mul(a));
66 }
67
68 return basicColor;
69 }
```

4 Probleme bei der Bearbeitung

Bei der Bearbeitung der Aufgabenstellung traten kaum Probleme auf, weil auf einer soliden Basis aufgebaut werden konnte.