Protokol Übung 03 - Beleuchtung I

Johann Roehl, Bodo Wissemann, Philip Zuschlag

1 Zum Inhalt dieses Protokols

Das Protokol behandelt die Implementierung des reflektierenden Materials sowie des Schatten-Werfens der verschiedenen Objekte. Hierfür wurden folgende Klassen neu implementiert:

- Tracer
- ReflectiveMaterial

Weiterhin wurden einige Klassen überarbeitet und um benötogte Felder ergänzt. Die GUI wurde ebenfalls angepasst.

2 Schatten

Die Klasse Light wurde um ein bool'sches Attribut *caststShadow* erweitert, welches nun allen Licht-Klassen beim Aufruf übergeben wird. Das *AmbientLight* ist das einzige Licht, welches keinen Schatten wirft. Zum errechnen, ob ein Schatten geworfen wird, wird in den *AmbientLight* der Licht-Klassen ein *Schatten-Ray* erzeugt. Schneidet dieser auf seinem Weg vom jeweiligen Punkt zur Lichtquelle ein Objekt, so wird ein Schatten geworfen. In der Klasse *PointLight* musste zudem der Fall, dass das Point-Light zwichen einem Objekt und dem jeweiligen Punkt liegt, beachtet werden.

```
public boolean illuminates(Point3 p, World world) throws IllegalArgumentException {
    if (p == null) {
        throw new IllegalArgumentException("The point3 cannot be null!");
    }

    if (castsShadows) {
        final Ray r = new Ray(p, directionFrom(p));

        final double tMax = r.tOf(pl);// t der Lichtquelle
        final double tMin = 0.00001;
        double t2 = 0;
        for (final Geometry g : world.objs) {

            final Hit h = g.hit(r);
            if (h != null) {

                t2 = h.t;
            }
            if (t2 >= tMin && t2 <= tMax) {

                return false;

            }
        }
    }
    return true;
}
```

Abbildung 1: *Die Methode Illuminates der Klasse PointLight*

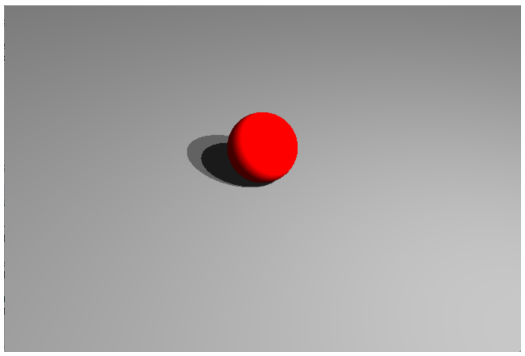

Abbildung 2: *Schattenüberlagerungen bei DirectionalLight und PointLight*

3 Das reflektierende Material

Eine neue Klasse *ReflectiveMateriel* wurde erstellt, welche neben Fraben für die diffuse Reflektion und den Glanzpunkt eine dritte Farbe für die Reflektion hat. Die Frabe der Reflektion werden in der Berechnung der Farben für jeden Pixel des Objektes am Ende hinzu multipliziert, nachdem sie rekursiv im Tracer errechnet wurden. Im Tracer war es sehr wichtig, die Rekursionstiefe zu beschränken, um ggf auftretende *StackOverflowExceptions* aufgrund unendliche reflektierter Starhlen vor zu beugen.

```
final Vector3 e = (hit.ray.direction.mul(-1.0)).normalized();
for (final Light light : world.lights) {

    if (light.illuminates(hitPoint, world)) {

        final Vector3 lightVector = light.directionFrom(hitPoint).normalized();

        final Vector3 reflectedVector = lightVector.normalized().reflectedOn(hit.n);

        final double max = Math.max(0.0, lightVector.dot(hit.n));
        final double maxSP = Math.pow(Math.max(0.0, reflectedVector.dot(e)),
                this.exponent);

        returnColor = returnColor.add(light.color.mul(this.diffuse).mul(max)).
                add(light.color.mul(this.specular).mul(maxSP));
    }
}

return returnColor.add(reflectionColor.mul(tracer.reflectedColors(new Ray(hitPoint,
        hit.ray.direction.add(hit.n.mul(factor)).normalized()))));
```

Abbildung 3: *Addition des reflektierten Lichtes in der Methode colorFor der Klasse ReflectiveLight*

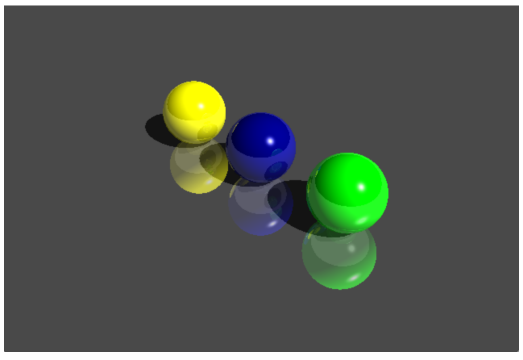

Abbildung 4: *Reflektionen mehrerer Objekte auf einer Plane*