

# PHYSICALLY BASED RENDERING, REPORTS

*Student:*

Andrea DISTLER, 130269

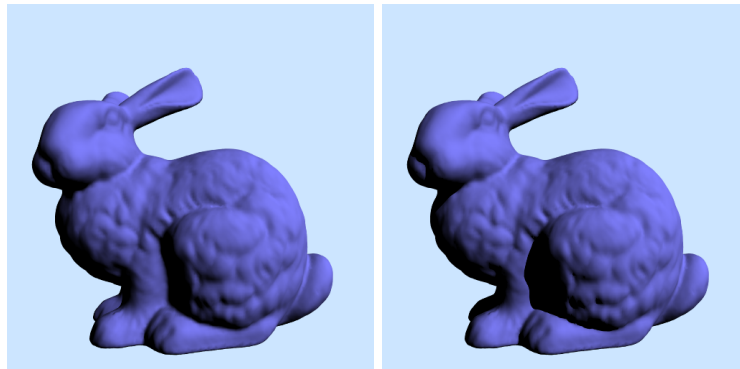
*Teacher:*

Jeppe FRISVALD

March 18, 2013

## Contents

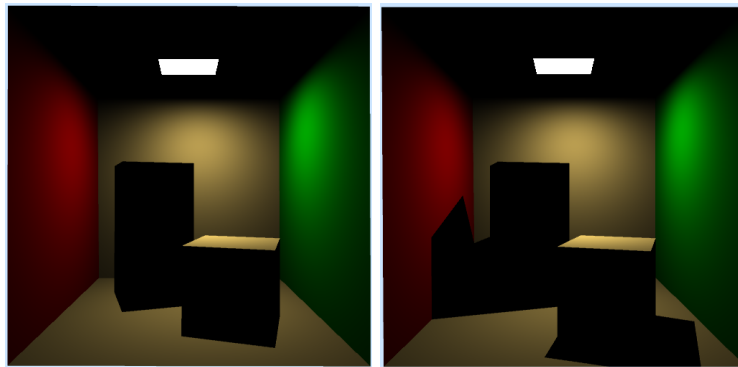
<b>1</b>	<b>Report Exercise 1</b>	<b>2</b>
<b>2</b>	<b>Report Exercise 2</b>	<b>5</b>



(a) Without shadows: 5s

(b) With shadows: 8s

Figure 1: Bunny.obj, Tris: 69451, 36 samples, 1 directional light, Lambertian shader



(a) Without shadows: 0.3s

(b) With shadows: 0.5s

Figure 2: Cornellbox.obj and CornellBlocks.obj, Tris: 36, 4 samples, 1 area light, Lambertian shaders

## 1 Report Exercise 1

- Implemented a directional light with shadows
- Implemented an area light with shadows

Listing 1 : Directional.cpp

```

1 bool Directional::sample(const Vec3f& pos, Vec3f& dir, Vec3f& L)
  const
2 {
3     dir = -light_dir;
4     L = emission;
5
6     // test for shadow
7     Ray shadowRay(pos, -light_dir);
8     bool inShadow = false;
9
10    if (shadows)
11        inShadow = tracer->trace(shadowRay);
12
13    return !inShadow;
14 }

```

Listing 2 : Lambertian.cpp

```

1 Vec3f Lambertian::shade(Ray& r, bool emit) const
2 {
3     Vec3f rho_d = get_diffuse(r);
4     Vec3f result(0.0f);
5
6     // temp light direction and radiance
7     Vec3f lightDirection, radiance;
8     for (std::vector<Light*>::const_iterator it = lights.begin(); it
9         != lights.end(); it++)
10    {
11        if ((*it)->sample(r.hit_pos, lightDirection, radiance))
12        {
13            // output of Lambertian BRDF
14            Vec3f f = rho_d * M_1_PIf;
15
16            // directional light radiance
17            // f - scattered light radiance, radiance - current light
18            // radiance, last term: cosine cut off at 0
19            result += f * radiance * std::max(dot(r.hit_normal,
20                lightDirection), 0.0f);
21        }
22    }
23
24    return result + Emission::shade(r, emit);
25 }

```

Listing 3 : AreaLight.cpp

```

1 bool AreaLight::sample(const Vec3f& pos, Vec3f& dir, Vec3f& L)
  const
2 {
3     // Get geometry info
4     const IndexedFaceSet& geometry = mesh->geometry;
5     const IndexedFaceSet& normals = mesh->normals;
6
7     // averaged light position
8     Vec3f lightPosition = Vec3f(0.0f);
9     // averaged normals
10    Vec3f lightNormal = Vec3f(0.0f);
11    // emission summed up from all faces
12    Vec3f emission = Vec3f(0.0f);

```

```

14 // iterate over all faces
15 for (int i = 0; i < geometry.no_faces(); i++)
16 {
17     // get the center of the face
18     Vec3i face = geometry.face(i);
19     Vec3f v0 = geometry.vertex(face[0]);
20     Vec3f v1 = geometry.vertex(face[1]);
21     Vec3f v2 = geometry.vertex(face[2]);
22     Vec3f faceCenter = v0 + (v1 - v0 + v2 - v0) * 0.5f;
23
24     // combine light position
25     lightPosition += faceCenter;
26
27     // average normals
28     lightNormal += (normals.vertex(face[0]) + normals.vertex(face
29         [1]) + normals.vertex(face[2])) / 3;
30
31     // add emission
32     emission += mesh->face_areas[i] * get_emission(i);
33 }
34
35 // average light position
36 lightPosition /= geometry.no_faces();
37
38 lightNormal.normalize();
39
40 // get light direction and distance to light
41 Vec3f lightDirection = lightPosition - pos;
42 float lightDistance = length(lightDirection);
43
44 // set area light direction, normalize
45 dir = lightDirection / lightDistance;
46
47 // set radiance
48 L = emission * std::max(dot(-dir, lightNormal), 0.0f) / (
49     lightDistance * lightDistance);
50
51 // trace for shadows
52 bool inShadow = false;
53 if (shadows)
54 {
55     Ray shadowRay(pos, dir);
56     shadowRay.tmax = lightDistance - 0.1111f;
57     inShadow = tracer->trace(shadowRay);
58 }
59
60 return !inShadow;
61 }

```

## 2 Report Exercise 2

The input parameters for the sun sky light are theta, and phi, together creating the solar position, and the turbidity (how much light is scattered due to dirt in the atmosphere, using empirical values - Preetham). Using Preetham's paper, theta and phi are calculated from the latitude, declination, julian day of the year and the time of the day, as well as some constants from Preetham's paper. The code for this can be found in listing 4.

The model used for calculating the sun's intensity is calculated as shown in listing 5. The sun covers a solid angle of  $2\pi$  degrees<sup>2</sup> ( $\Rightarrow$  the whole hemisphere).

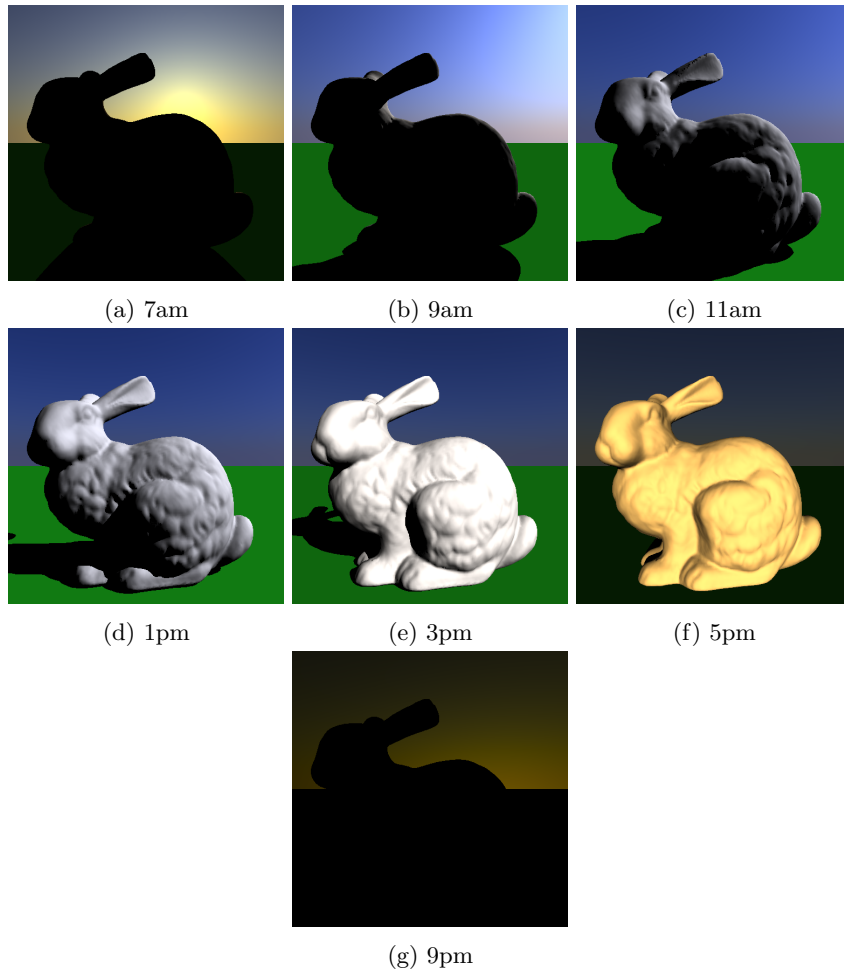


Figure 3: Bunny.obj and plane, Tris: 70.000, 1 sample, 1 skylight, Lambertian shaders  $\Rightarrow$  approx. 3s per picture

Listing 4 : RenderEngine::init\_tracer()

```

1  if(use_sun_and_sky)
    {
3      // Use the Julian date (day_of_year), the solar time (
        time_of_day), the latitude (latitude),
        // and the angle with South (angle_with_south) to find the
        direction toward the sun (sun_dir).

5      // hard coded numbers are from Preetham et al.'s A Practical
        Analytical Model for Daylight, SIGGRAPH 1999
7      float declination = 0.4093 * sin(2 * M_PIf * (day_of_year - 81)
        / 368);
        float theta = M_PIf * 0.5f - asin(sin(latitude) * sin(
            declination) -
9            cos(latitude) * cos(declination) * cos(M_PIf * time_of_day /
            12));
        float phi = atan(-(cos(declination) * sin(M_PIf * time_of_day /
            12)) /
11        (cos(latitude) * sin(declination) - sin(latitude) * cos(
            declination) * cos(M_PIf * time_of_day / 12)));

13        sun_sky.setSunTheta(theta);
        sun_sky.setSunPhi(phi);
15        sun_sky.setTurbidity(turbidity);
        sun_sky.init();
17        tracer.set_background(&sun_sky);
    }

```

Listing 5 : PreethamSunSky::sample(..)

```

bool PreethamSunSky::sample(const Vec3f& pos, Vec3f& dir, Vec3f& L)
    const
2  {
    dir = const_cast<PreethamSunSky*>(this)->getSunDir();

4      float area = 1;
        float solid_angle = 2 * M_PI;
        float cos_theta = dot(Vec3f(0, 1, 0), dir);

8      // * 0.00001f to convert to the right unit (cd/m^2)
10     L = const_cast<PreethamSunSky*>(this)->sunColor() / (area *
        solid_angle * cos_theta) * 0.00001f;

12     // test for shadow
        Ray shadowRay(pos, dir);
14     bool inShadow = false;

16     if (shadows)
        inShadow = tracer->trace(shadowRay);
18
        return !inShadow;
20 }

```