5.3 Monte Carlo Integration

Write a little program in your favourite programming language to estimate the value of π using rejection sampling. Randomly generate $N = 1.000.000$ points in the range $[-1, 1]$ and count which fraction of them end up inside a unit-circle. From this fraction, you can compute the surface area of the unit-circle, which is π.
Only submit your sourcecode
Hint: See here to learn how to include source code in LaTeX: `https://en.wikibooks.org/wiki/LaTeX/Source_Code_Listings`

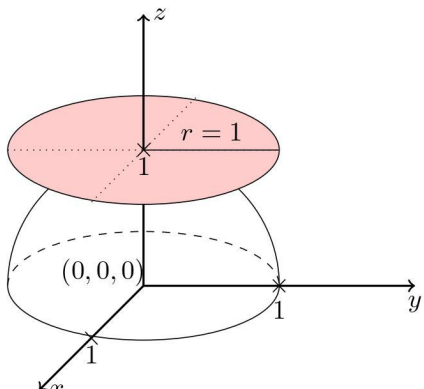

Figure 2: A disc on the $z = 1$ plane. Your task is to compute the solid angle covered by this disk for an observer at the origin $(0, 0, 0)$.

5.4 From Angles to Solid Angles

Write a little program in your favourite programming language to estimate the solid angle of a unit-circle that is parallel to the xy-plane with $z = 1$. The unit circle is shown in Figure 2. You have to uniformly sample directions on the hemisphere, intersect those directions with the $z = 1$ plane and then count the fraction of the intersections that end up inside the circle.
Hint: Given two random variables ξ_1, ξ_2 you can compute uniformly distributed points on the hemisphere in spherical coordinates like this:

$$\theta = \arccos(\xi_1)$$
$$\phi = 2\pi\xi_2$$

5.5 Implement a Path Tracer

This time, we extend our Whitted-style ray tracer to a path tracer with full global illumination.

a) In order to capture the entire global illumination, we need to create random light-transport paths in the scene and compute their final contribution to the image. At every intersection, we still evaluate the local rendering equation, but for all following intersections, we multiply a `throughput` factor on all the radiance L we gather to account for the attenuation caused by previous intersections.

A code template for the path tracer is already given in

```
Color RayTracer::pathIntegrator(const Scene& scene, const Ray& cameraRay) const
```

Your task is to sample an outgoing ray direction ω_i using the `Sampler` `class`, transform it to world space, set it as the new `ray` (starting at the current intersection point), and multiply

$$\frac{f_r(\omega_i, x, \omega_o)\cos\theta_i}{p(\omega_i)}$$

onto the `throughput`. Here, $p(\omega_i)$ is the *probability density* of the sampled direction. You can find the matching function in the `Sampler` `class`.

b) Another new feature supported by option to have area lights. Now, meshes can emit a constant amount of radiance L_e into the upper hemisphere.

We can either compute their contributions to the pixel value by tracing random light-transport paths through the scene or we can also, like for the point light, sample a direct connection to them. Unlike the point light, this requires choosing a random location on the mesh first. That part is already given in the template code

```
std::pair<Color, Point3D> Light::Area::sampleLi(Point3D receiver, Point2D
    sample) const
```

By sampling random locations on the mesh, we're computing an integral over differential area dA. However, our path tracer requires us to use differential solid angle $d\omega_i$.

Your task is to multiply the emitted radiance L_e by the conversion factor $\frac{d\omega_i}{dA}$ and the total area of the mesh A in order to compute the result

$$\frac{L_e(y, -\omega_i)}{p(\omega_i \mid x)} = \frac{L_e(y, -\omega_i) A \, d\omega_i}{dA}.$$

Here, y is the sampled point on the emitter and x is the current intersection point.

To check if you computed the correct result here, check if your path tracer computes the same result (with generally less noise) when you set `sampleAreaLights = true`.

You can see an example of the updated framework in Figure 3. You can fly through the scene using W, A, S, D or the arrow keys (S and D look around and Q and E move left and right). To look up or down use Insert and Delete. To move up or down, use Space and X.
If you want to show the 3D mesh on the side, or change the scene's meshes or materials simply edit the `main.cpp`. Otherwise, you again only need to implement things in the `exercise05.cpp` file.

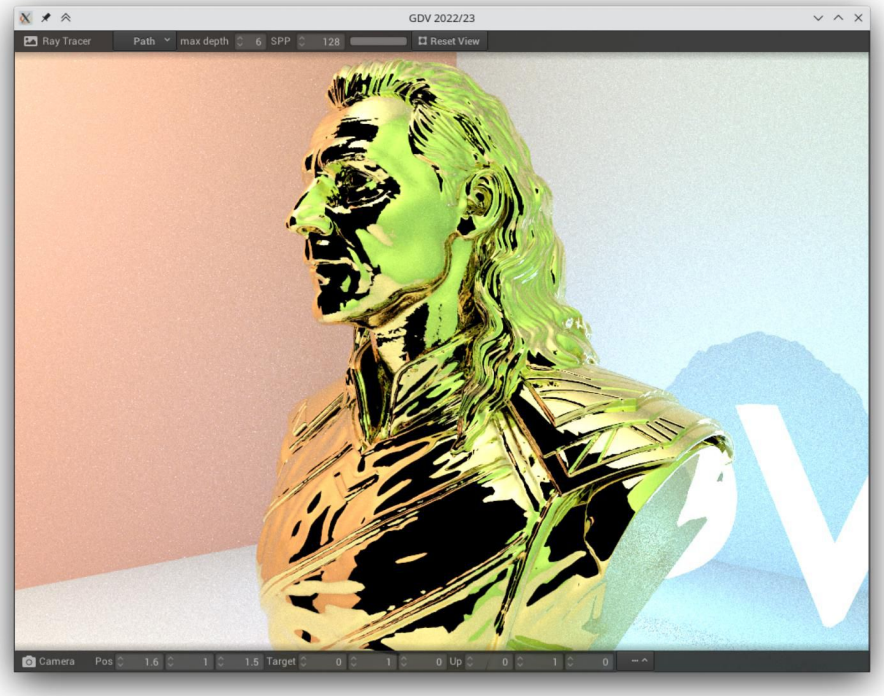

Figure 3: The result of using the path tracer to render the given scene with 128 samples per pixel (the camera position has been moved).