ECE 6360 Parallel and Heterogeneous Computing

Raytracing – Parallel Path Tracing

Rubric

repository	_____ **/ 10**
CMake repository compiles (gcc/g++)	_____ / required
Organization	_____ / 10
path tracing implementation	_____ **/ 50**
random sampling for diffuse reflections	_____ / 15
scattering function for spheres	_____ / 20
path tracing algorithm	_____ / 15
parallelism	_____ **/ 40**
subdivide workload across arbitrary threads	_____ / 15
report on parallel speedup	_____ / 25

Helpful Pseudocode

- **Feel free to ask about any of these algorithms in the General chat on Teams**

Path Tracing (general approach)

```
while ( rendering ) {
  for each pixel:
    r = GenerateRay(...)
    path_color = [0, 0, 0]
    path_scattering = [1, 1, 1]
    for each bounce:
      t, entity = CalculateClosestHit(r, ...)
      r = entity->Scatter(...)
      path_color += entity->emission * path_scattering
      path_scattering *= entity->scattering;
}
```

Scattering

```
Scatter(...) function:
  p = r(p)
  n = normal()
  new_ray = SampleHemisphere(n, ...)
  cos(θ) = dot(new_ray, n)
  return emission, scattering * cos(θ)
```

Sample Hemispheres

You'll have to write a function that randomly samples a hemisphere. As input, you'll need the surface normal $\boldsymbol{n}$ and the solid angle α that you want to sample (for a hemisphere $\alpha = \pi$). You will also need two random numbers $x_0, x_1 \in [0, 1)$.

You can uniformly sample the surface of a unit sphere using Archimedes' principle (see lecture). First, use the random numbers to calculate a sample on a unit cylinder in cylindrical coordinates (θ, z):

$$z = x_0(1 - \cos\alpha) + \cos\alpha \qquad\qquad \theta = x_1 2\pi$$

Then project these spherical coordinates onto a unit sphere in spherical coordinates (θ, ϕ) – since you already have θ you can calculate $\phi = \cos^{-1} z$. Converting these spherical coordinates into Cartesian coordinates gives you a vector that is within the solid angle α of the z-axis:

$$\boldsymbol{v}' = \begin{bmatrix} \cos\theta \sin\phi \\ \sin\theta \sin\phi \\ \cos\phi \end{bmatrix}$$

This vector v provides a new random ray orientation that lies within the specified solid angle ($\alpha = \pi$ for a hemisphere) of the z-axis. Now you can align the z-axis with the surface normal by calculating a rotation matrix R to transform $\boldsymbol{v}'$.

Calculate the axis and rotation angle β needed to rotate the z-axis to align with $\boldsymbol{n}$. This can be done using the cross product:

$$\boldsymbol{a} = \boldsymbol{n} \times \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

where the rotation axis is $\boldsymbol{s} = \frac{\boldsymbol{a}}{||\boldsymbol{a}||}$ and $\cos\beta = \boldsymbol{s} \cdot \boldsymbol{n}$.

$$\boldsymbol{R} = \begin{bmatrix} s_x^2(1 - \cos\beta) + \cos\beta & s_x s_y(1 - \cos\beta) - s_z \sin\beta & s_x s_z(1 - \cos\beta) + s_y \sin\beta \\ s_x s_y(1 - \cos\beta) + s_z \sin\beta & s_y^2(1 - \cos\beta) + \cos\beta & s_y s_z(1 - \cos\beta) - s_x \sin\beta \\ s_x s_z(1 - \cos\beta) - s_y \sin\beta & s_y s_z(1 - \cos\beta) + s_x \sin\beta & s_z^2(1 - \cos\beta) + \cos\beta \end{bmatrix}$$

The final sample direction for your new ray is:

$$\boldsymbol{v} = \boldsymbol{R}\boldsymbol{v}'$$