MPCS 51087
Project 2
GPU Ray Tracing with CUDA

Winter 2024

Milestone 1: Serial version due Sun, Feb 4 at 6pm.

1 Introduction to Ray Tracing

Ray tracing is a powerful method for rendering three-dimensional objects with complex light interactions. Many other physical phenomena are also analogous to ray tracing – for example, simulating neutrons passing through matter. Figure 1 shows the basic idea for a reflective object. An observer (the eyeball) is viewing an object through a window (the rectangle). The object is illuminated by a light source, which is emitting rays of light (the arrows) in many directions. The observer will see rays that reflect off of the object.

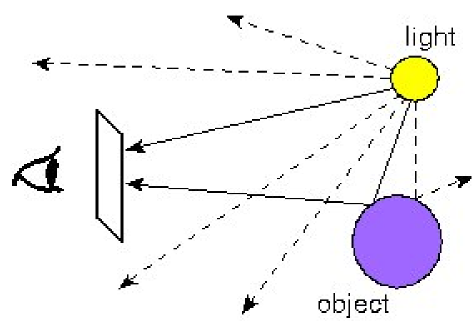

Figure 1: Illustration of Ray-Tracing. Image credit: [**??**]

Our task is to render the image seen by the observer through the window. To do so, the simulation can do one or both of the following:

- Simulate light rays starting at the light source and ending at the observer.
- Simulate light rays starting from the observer and going backwards to the light source.

We will implement the latter scheme in this assignment. We will simulate a reflective sphere in black-and-white, illuminated by a single light source. A serial version can be implemented in about 100 lines of code.

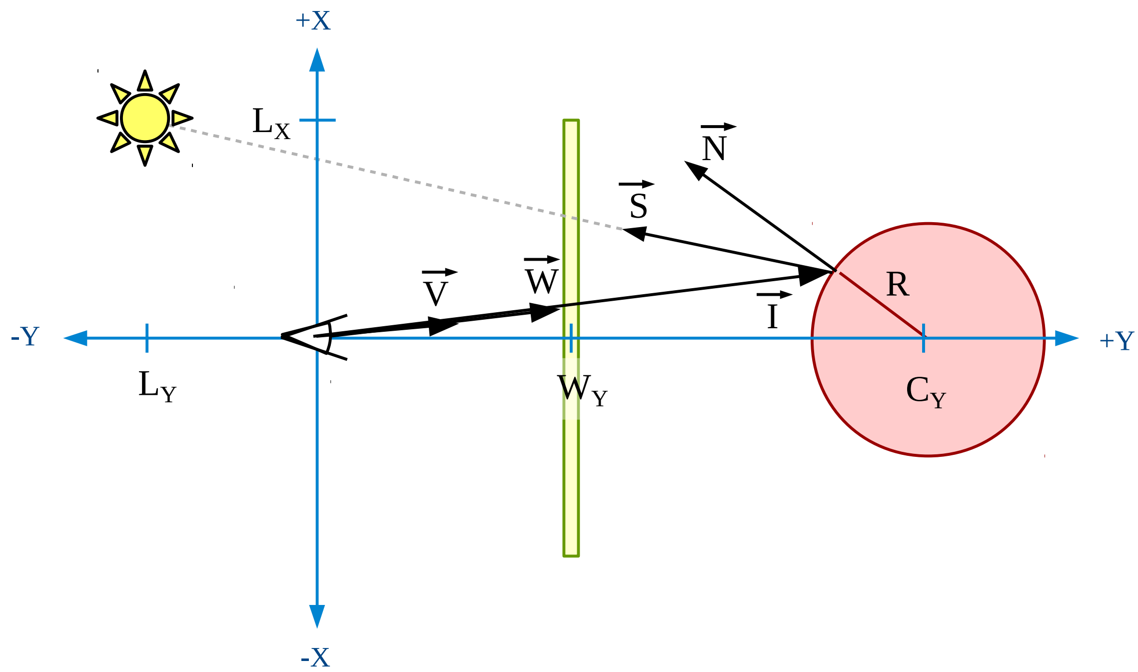

Figure 2: 2D Diagram for Ray-Tracing

2 Vector Notation

Solving the problem involves the use of 3D vectors. A vector $\vec{V}$ has scalar components,$\vec{V} = (V_X, V_Y, V_Z)$. Adding or subtracting two vectors yields a vector, $\vec{V} - \vec{U} = (V_X - U_X, V_Y - U_Y, V_Z - U_Z)$. Multiplying a scalar and a vector yields a scalar, $t\vec{V} = (tV_X, tV_Y, tV_Z)$. Dividing a vector by a scalar yields a scalar, $\vec{V}/t = (V_X/t, V_Y/t, V_Z/t)$. The dot product (a multiplication between two vectors) yields a scalar, $\vec{V} \cdot \vec{U} = V_X U_X + V_Y U_Y + V_Z U_Z$. The norm (or magnitude or length) of the vector is a scalar defined by $|\vec{V}| = \sqrt{V_X^2 + V_Y^2 + V_Z^2}$.

3 Theory

This section describes the theory used in our implementation. The implementation itself is brief and is described in the algorithms below, so you may skip this section as needed.

Figure 2 shows a 2D cross-section of the problem with the information we need. The observer is at the origin and faces the positive y-direction. The sphere is located at $\vec{C} = (C_X, C_Y, C_Z)$ and has radius R. The window is parallel to the (x,z)-plane at $y = W_Y$ and has bounds $-W_{max} < W_X < W_{max}$ and $-W_{max} < W_Z < W_{max}$. In the simulation, we will represent the window as an $n \times n$ grid, G. The light source is located at $\vec{L} = (L_X, L_Y, L_Z)$.

Our task is to simulate many rays originating from the observer (so-called "view rays") with randomly-selected directions. The steps are as follows:

- **Select the direction of view ray ($\vec{V}$).** Let $\vec{V}$ be a unit vector representing the direction of the view ray. In spherical coordinates, we will randomly select its component angles (θ, ϕ).

Then we will get $\vec{V}$ in Cartesian coordinates:

$$
\begin{aligned}
V_X &= \sin\theta\cos\phi \\
V_Y &= \sin\theta\sin\phi \\
V_Z &= \cos\theta
\end{aligned}
$$

One important point to note is that we cannot sample both ϕ and θ from uniform random distributions between 0 and 2π without biasing the distribution. To sample the unit sphere correctly, we will sample only ϕ from a uniform distribution between 0 and 2π. We will then sample the *cosine* of θ from a uniform distribution between -1.0 and 1.0. E.g.,

Algorithm 1 Direction Sampling Algorithm

Sample ϕ from uniform distribution $(0, 2\pi)$
Sample $\cos(\theta)$ from uniform distribution $(-1, 1)$
$\sin(\theta) = \sqrt{1 - \cos^2(\theta)}$
$V_X = \sin(\theta)\cos(\phi)$
$V_Y = \sin(\theta)\sin(\phi)$
$V_Z = \cos(\theta)$

Also note that we cannot sample V_X, V_Y, and V_Z directly from a uniform distribution $(-1, 1)$ in all three Cartesian dimensions and then normalize, as this would generate a biased distribution.

- **Find the intersection of the view ray with the window ($\vec{W}$).** Knowing that the window is at W_Y, the window's point-of-intersection with the view ray is given by the vector $\vec{W}$:

$$\vec{W} = \frac{W_Y}{V_Y}\vec{V}$$

  If the view ray is outside the window ($|W_X| > W_{max}$ or $|W_Z| > W_{max}$), we reject it and chose a new $\vec{V}$.

- **Find the intersection of view ray with sphere ($\vec{I}$).** Let $\vec{I}$ be the sphere's point-of-intersection with the view ray. To find $\vec{I}$, we solve the following system of equations:

$$
\begin{aligned}
\vec{I} &= t\vec{V} \\
\left|\vec{I} - \vec{C}\right|^2 &= R^2
\end{aligned}
$$

  These are the equations of the view ray and the sphere, respectively. Solving for t yields:

$$t = \left(\vec{V} \cdot \vec{C}\right) - \sqrt{\left(\vec{V} \cdot \vec{C}\right)^2 + R^2 - \vec{C} \cdot \vec{C}}$$

  which can be back-substituted to get $\vec{I}$. If t does not have a real solution ($(\vec{V} \cdot \vec{C})^2 + R^2 - \vec{C} \cdot \vec{C} < 0$), then view ray does not intersect the sphere and we choose a new $\vec{V}$.

- **Find the observed brightness of the sphere (b).** Next, we want to find the brightness of the sphere that is observed at $\vec{I}$. To do so , we:

  - *Find the unit normal vector ($\vec{N}$).* The unit normal vector $\vec{N}$ is perpendicular to the sphere's surface at $\vec{I}$.

$$\vec{N} = \frac{\vec{I} - \vec{C}}{\left|\vec{I} - \vec{C}\right|}$$

- *Find the direction to the light source ($\vec{S}$).* The direction to light source (sometimes called the "shadow ray") is represented by the unit vector $\vec{S}$.

$$\vec{S} = \frac{\vec{L} - \vec{I}}{\left|\vec{L} - \vec{I}\right|}$$

- *Find the brightness (b).* The brightness can be found from $\vec{S}$ and $\vec{N}$ using "Lambertian shading".

$$b = \begin{cases} 0 & \vec{S} \cdot \vec{N} < 0 \\ \vec{S} \cdot \vec{N} & \vec{S} \cdot \vec{N} \geq 0 \end{cases}$$

- **Add the brightness to the window's grid.** We find (i, j) such that $\vec{G}(i, j)$ is the position of $\vec{W}$ on the window's grid G and let:

$$G(i, j) = G(i, j) + b$$

4 Implementation

Algorithm 2 describes a ray-tracing implementation. As described above, the observer is at the origin and is facing the positive-y direction. The sphere is located at $\vec{C} = (C_X, C_Y, C_Z)$ and has radius R. The light source is located at $\vec{L} = (L_X, L_Y, L_Z)$. The window is parallel to the (x,z)-plane at $y = W_Y$, has bounds $-W_{max} < W_X < W_{max}$ and $-W_{max} < W_Z < W_{max}$, and is represented by an $n \times n$ grid, G.

Algorithm 2 Ray Tracing Algorithm

1: allocate $G[1 \dots n][1 \dots n]$ ▷ The window is represented on the grid G
2: $G[i][j] = 0$ for all (i, j)
3: **for** $n = 1 \dots N_{rays}$ **do**
4:     **repeat**
5:         Sample random $\vec{V}$ from unit sphere ▷ Algorithm 1
6:         $\vec{W} = \frac{W_Y}{V_Y}\vec{V}$ ▷ The intersection of the view ray and the window
7:     **until** $|W_X| < W_{max}$ **and** $|W_Z| < W_{max}$ **and** $\left(\vec{V} \cdot \vec{C}\right)^2 + R^2 - \vec{C} \cdot \vec{C} > 0$
8:     $t = \left(\vec{V} \cdot \vec{C}\right) - \sqrt{\left(\vec{V} \cdot \vec{C}\right)^2 + R^2 - \vec{C} \cdot \vec{C}}$
9:     $\vec{I} = t\vec{V}$ ▷ The intersection of the view ray and the sphere
10:     $\vec{N} = \frac{\vec{I} - \vec{C}}{\left|\vec{I} - \vec{C}\right|}$ ▷ The unit normal vector at $\vec{I}$
11:     $\vec{S} = \frac{\vec{L} - \vec{I}}{\left|\vec{L} - \vec{I}\right|}$ ▷ The direction of the light source at $\vec{I}$
12:     $b = \textsc{max}\left(0, \vec{S} \cdot \vec{N}\right)$ ▷ The brightness observed at $\vec{I}$
13:     find (i, j) such that $G(i, j)$ is the gridpoint of $\vec{W}$ on G
14:     $G(i, j) = G(i, j) + b$ ▷ Add brightness to grid (must be thread safe!)
15: Output grid G to file

Note that in Algorithm 2, all rays are independent from one another, except for the fact that they are all tallying their results to a single common grid (Line 14). As the rays are randomized, it may be the case that multiple rays try to update the value at the same grid location at the same time, resulting in undefined behavior. Therefore, some sort of synchronization will be necessary to ensure correctness.

Figure 3 shows a sample image where $\vec{L} = (4, 4, -1)$, $W_Y = 10$, $W_{max} = 10$, $C = (0, 12, 0)$, $R = 6$.

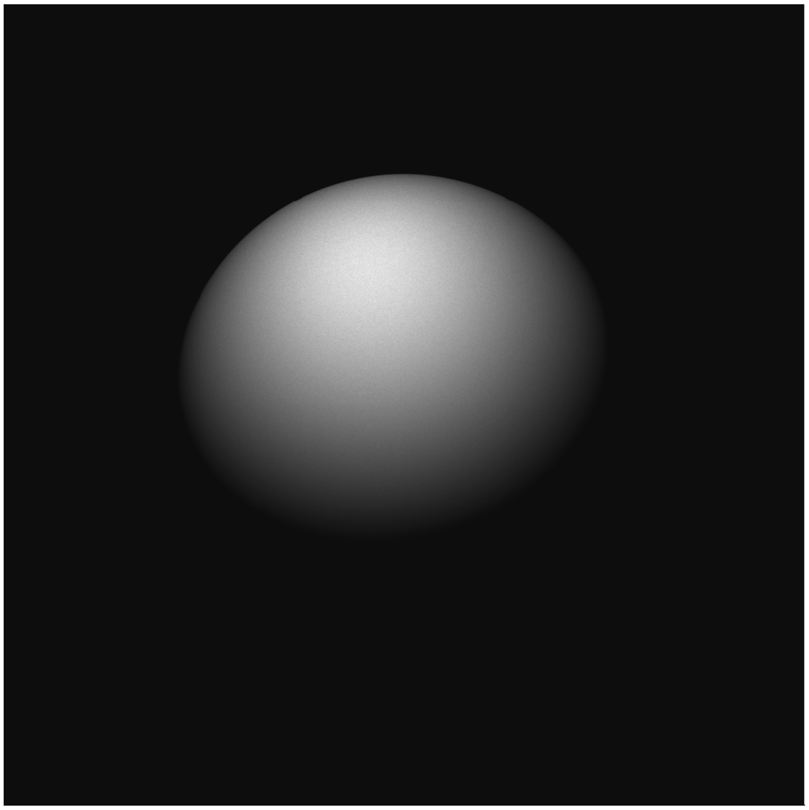

Figure 3: Ray-Traced Render of a Sphere Illuminated from Top Left