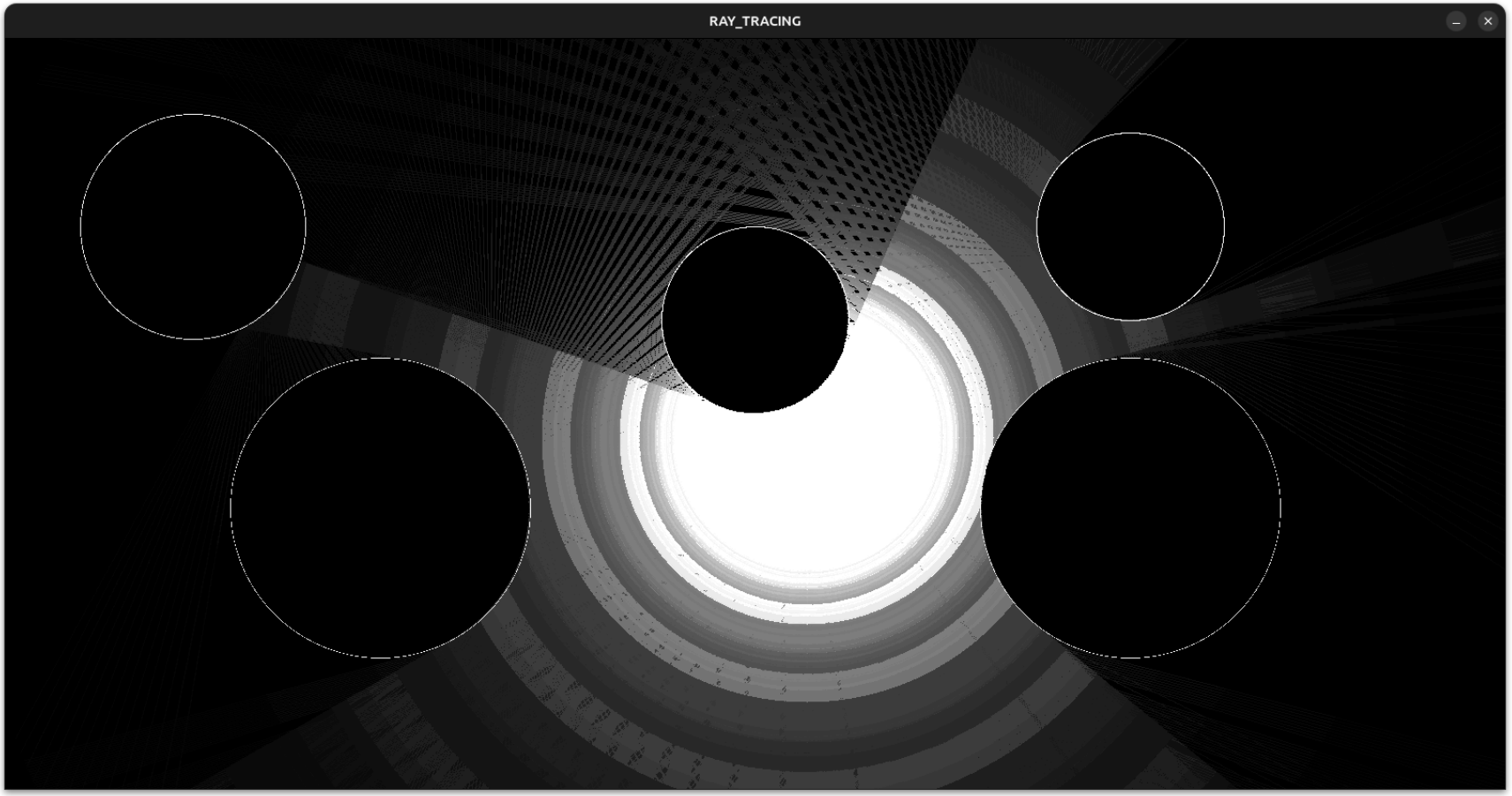
🌟 Ray Tracing with Specular Reflections

> **Real-time light simulation using SDL2 and C**



Still example of rays bouncing off reflective circles.

Still example of rays not bouncing off reflective circles. NOTICE THE MAJOR PERFORMANCE DIFFERENCE.

🔬 Mathematical Foundation

➤ Ray Representation

Each ray is defined by:

- **Origin**: (x_s, y_s)
- **Direction**: $\vec{d} = (dx, dy)$
- **Intensity**: I
- **Bounce count**: b

➤ Ray Propagation

Rays follow:

$$x_{t+1} = x_t + dx \cdot \Delta t$$
$$y_{t+1} = y_t + dy \cdot \Delta t$$

➤ Circle Intersection

A ray at point (x, y) intersects a circle (x_c, y_c, r) if:

$$(x - x_c)^2 + (y - y_c)^2 \leq r^2$$

➤ Specular Reflection

When a ray hits a circular mirror:

1. **Surface Normal**:

$$\vec{n} = \left(\frac{x - x_c}{r}, \frac{y - y_c}{r} \right)$$

2. **Incident Vector**:

$$\vec{i} = (dx, dy)$$

   (normalized to unit vector)

3. **Reflected Vector**:

$$\vec{r} = \vec{i} - 2(\vec{i} \cdot \vec{n})\vec{n}$$

   (then normalized)

➤ Intensity Attenuation

Light intensity diminishes based on:

1. **Distance attenuation**:

$$I_d = I_0 \cdot \frac{k}{d^2 + 1}$$

   where $k = 20000$ is the attenuation constant

2. **Reflection attenuation**:

$$I_{b+1} = I_b \cdot 0.8$$

🧱 Key Components

💡 1. Light Source

- Movable with mouse
- Emits `8000` rays
- Color: `0xffffffff` (white)

🟠 2. Circular Obstacles

- Perfect **mirror-like** circles
- Drawn as white outlines

🌀 3. Ray Tracing Algorithm

```
void FillRays(SDL_Surface *surface, struct Ray rays[],
              struct Circle objects[], int num_objects, Uint32 baseColor)
{
    for (all rays) {
        while (bounces < threshold) {
            while (!boundary && !hit) {
                x += dx;
                y += dy;
                dist_sq = (x - start_x)^2 + (y - start_y)^2;
                intensity = base_intensity * 20000 / (dist_sq + 1);
                blend_pixel(surface, x, y, color, intensity);

                for (all objects) {
                    if (intersects_circle(x, y, object)) {
                        handle_reflection(&ray, x, y, object);
                        hit = true;
                        break;
                    }
                }
            }
        }
    }
}
```

🪞 4. Reflection Handling

```
void handle_reflection(struct Ray* ray, double hit_x, double hit_y, struct Circle circle)
{
```

```
    nx = (hit_x - circle.x)/circle.r;
    ny = (hit_y - circle.y)/circle.r;

    ilen = sqrt(ray->dx*ray->dx + ray->dy*ray->dy);
    ix = ray->dx / ilen;
    iy = ray->dy / ilen;

    dot = ix*nx + iy*ny;
    rx = ix - 2*dot*nx;
    ry = iy - 2*dot*ny;

    rlen = sqrt(rx*rx + ry*ry);
    ray->dx = rx/rlen;
    ray->dy = ry/rlen;

    ray->x_start = hit_x + ray->dx*0.01;
    ray->y_start = hit_y + ray->dy*0.01;
    ray->intensity *= 0.8;
    ray->bounce_count++;
}
```

⚙️ Physics Simulation

- **Energy loss per bounce**: `* 0.8`
- **Max bounces**: 2
- **Min intensity**: 0.1
- **Self-intersection avoidance**: offset by `0.01` units

🚀 Performance Optimizations

Feature	Value
Ray count	8000
Max steps per ray	2000
Max bounces	2
Intensity cutoff	`< 0.01`

🛠️ Build & Run

🔧 Dependencies

- SDL2

- GCC (or any C compiler)

🧪 Compile

```
gcc -o ray_tracing ray_tracing.c -lSDL2 -lm
```

🎮 Controls

- Move light source with mouse drag
- Close window to exit

🌱 Future Enhancements

- 🌈 **Diffuse reflections**
- 🔵 **Color absorption**
- 🪟 **Refraction effects**
- 🧵 **Texture mapping**
- 🗂️ **Spatial partitioning**

📄 License

🆓 **MIT License** Free for educational and personal use.
