COMP5411 Rendering Project
Lens Renderer

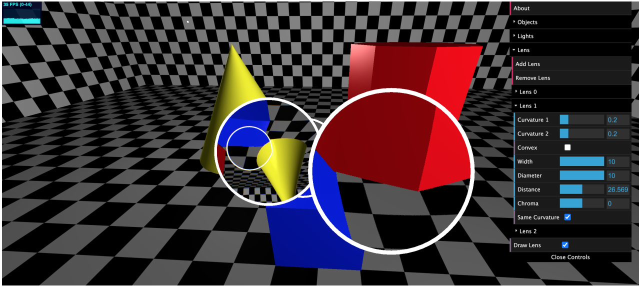
Anshuman Medhi
Aaron Si-yuan Wang

Group 21

Project Summary

Rendering Lenses

- Render a standard scene with objects and lighting, and include interactive magnifying lenses
- Simulate realistic refraction and dispersion of light to create all kinds of lens distortion
- Multipass rendering to handle multiple lens interaction



Curvature 1: 0.2
Curvature 2: 0.2
Convex: □
Width: 10
Diameter: 10
Distance: 26.569
Chroma: 0
Same Curvature: ☑
Draw Lens: ☑

Project Summary

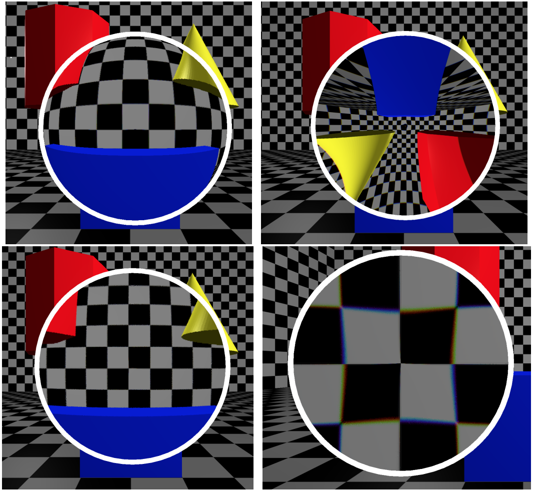


Figure: barrel, pincushion, and flat distortion, chromatic abberation

Features

Customizable Geometric Properties of Lens

- Radius of curvature of each surface
- Diameter of lens
- Thickness of glass
- Position of lens

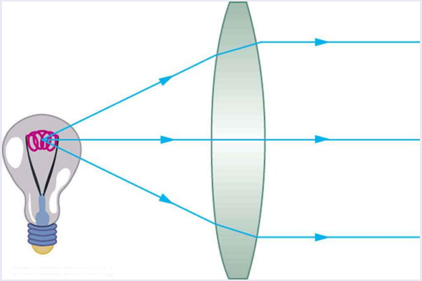
Customizable Scene Parameters

- Phong shading parameters for each object
- Position of point lights
- Number of lenses

Implementation

Lens Fragment Shader

1. Find the center of each spherical cap of the lens
2. Cast a ray through the lens onto the scene (a flat quad) refracting on the front and back surfaces



Implementation

Lens Fragment Shader

1. Each color is refracted separately to handle dispersion effect

$$r = \frac{R}{2} \qquad g = \frac{G}{2} \qquad b = \frac{B}{2}$$

$$y = \frac{2R + 2G - B}{6} \qquad c = \frac{2G + 2B - R}{6} \qquad v = \frac{2B + 2R - G}{6}$$

$$R = r + \frac{2v + 2y - c}{3} \quad G = g + \frac{2y + 2c - v}{3} \quad B = b + \frac{2c + 2v - y}{3}$$

Implementation

Multipass Rendering

1. Render the original scene into a `WebGLRenderTarget` texture t_0
2. For each lens i we create a new scene which consists of:
   - Orthographically projecting the previous texture t_{i-1} onto the next texture
   - Using a `CircleGeometry` with a custom shader material to distort a part of the previous texture t_{i-1}
   - Blending the previous texture and the distorted circle together
3. Step 2 repeats for every lens, except the last lens does not render to a texture, but directly to a screen
4. We ensure that lenses are rendered in depth-order so that lenses can 'see' each other