



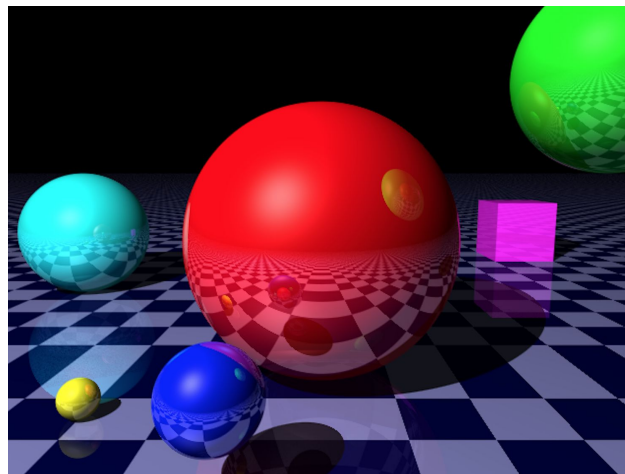
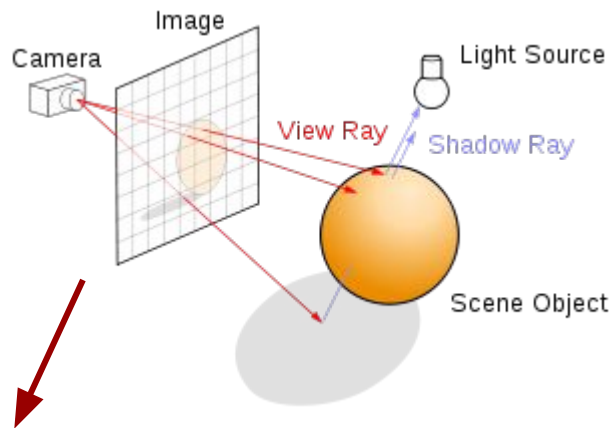
Ray Tracer

Capturing the Path of Light

Kiet Tran and Ojashvi Rautela

What is Ray Tracing?

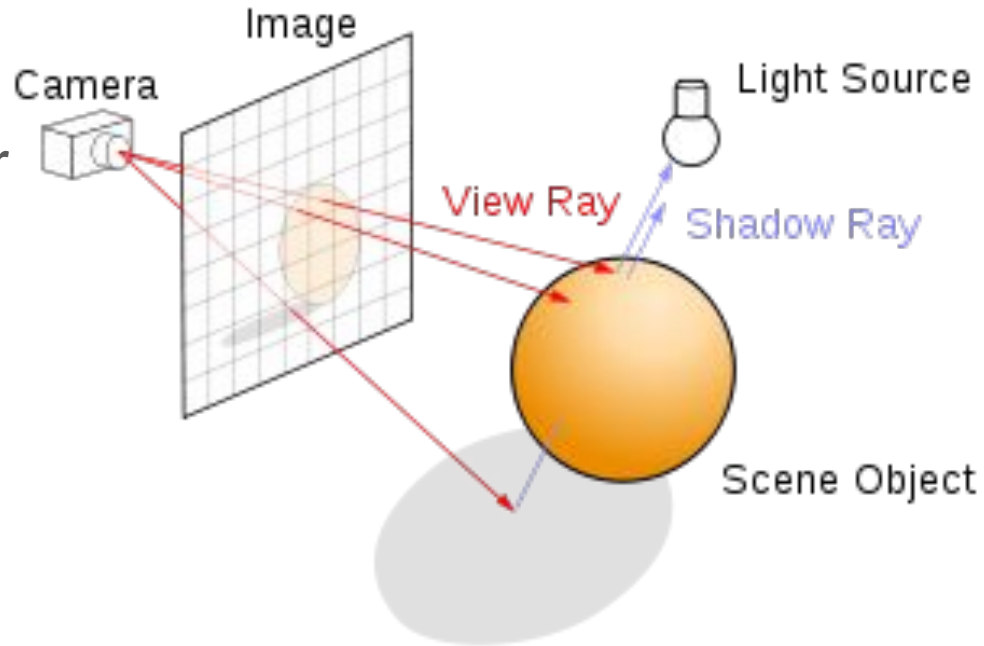
Simulating how light works in real life:
light source -> primary
object -> *physical*
interactions -> eye



What is Ray Tracing?

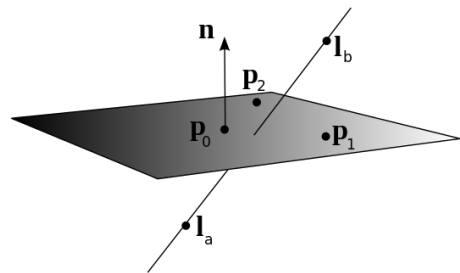
Physical interactions: shadows, reflection, refraction based on object properties such as **specular** or **diffusion** coefficients

For every pixel in the image plane, check for primary and secondary ray-object intersections



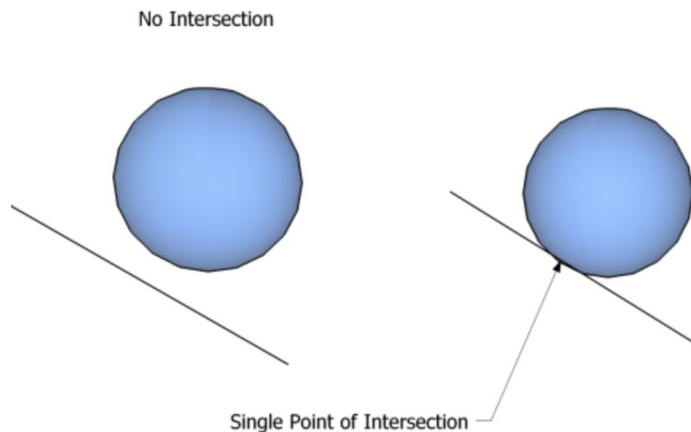
Important Math : Object-Ray interactions

- Plane Object and Ray (Line) intersections
 - Plane Eq. for a set of points p
 - $(p - p_0) \cdot n = 0$
 - Point on a line (light ray)
 - $\text{ray_origin} + (\text{ray_dir} * t)$
 - Essentially the equation of a line
 - $\text{ray_origin} = \text{intercept}; \text{ray_dir} = \text{slope}$
 - $t = \text{point of intersection}$
 - **Solve for t** = Substitute the eq. Of the line into the plane



Important Math : Object-Ray interactions

- Sphere Object and (Ray) Line intersections
 - Sphere Eg.
 - $x^2 + y^2 + z^2 = r^2$
 - Point on ray line
 - $\text{ray_origin} + (\text{ray_dir} * t)$
 - Solve for t = Substitute the eq. of the line into the plane and



Our Implementation : An Overview

- 1 Basic Scene**
 1. Vector, Color, Ray
 2. Camera, Light, Objects
 3. Ambient Light
- 2 Object Properties**
 1. Ray-Object interactions
 2. Reflectivity, Transparency
 3. Specular, Diffusion
- 3 Light Properties**
 1. Shadow
 2. Reflection
 3. Refraction
- 4 Anti-Aliasing**
 1. Averaging RGB components of n pixels around current
 2. n = depth

Program Structure (.cpp and .h files)



1. Reflection only
2. Reflection and Refraction

01 | main.cpp - create rays, render the scene via. anti-aliasing

02 | App.cpp - getColorAt() is the main *ray tracing* function

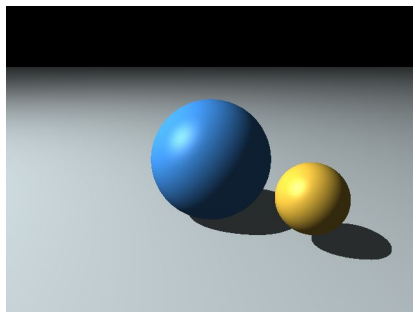
03 | getColorAt() - shadows, ambient, diffuse, reflection, refraction

04 | Other classes: simulate vectors, camera, light, rays, objects

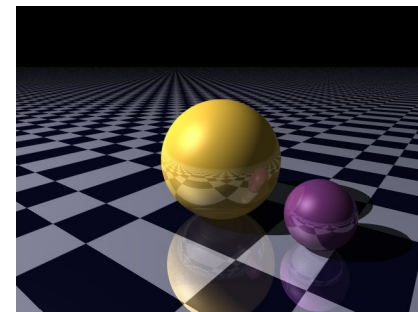
A wide-angle photograph of a high-altitude mountain range. The foreground is a smooth, snow-covered slope. In the middle ground, several jagged mountain peaks are visible, some covered in snow and others showing dark, rocky surfaces. The background shows more distant peaks under a sky with soft, white clouds. The overall color palette is dominated by whites, greys, and light blues.

Milestones

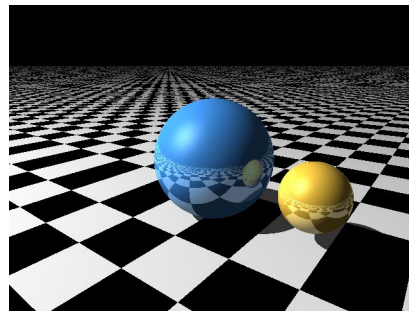
Rendered first
scene with objects



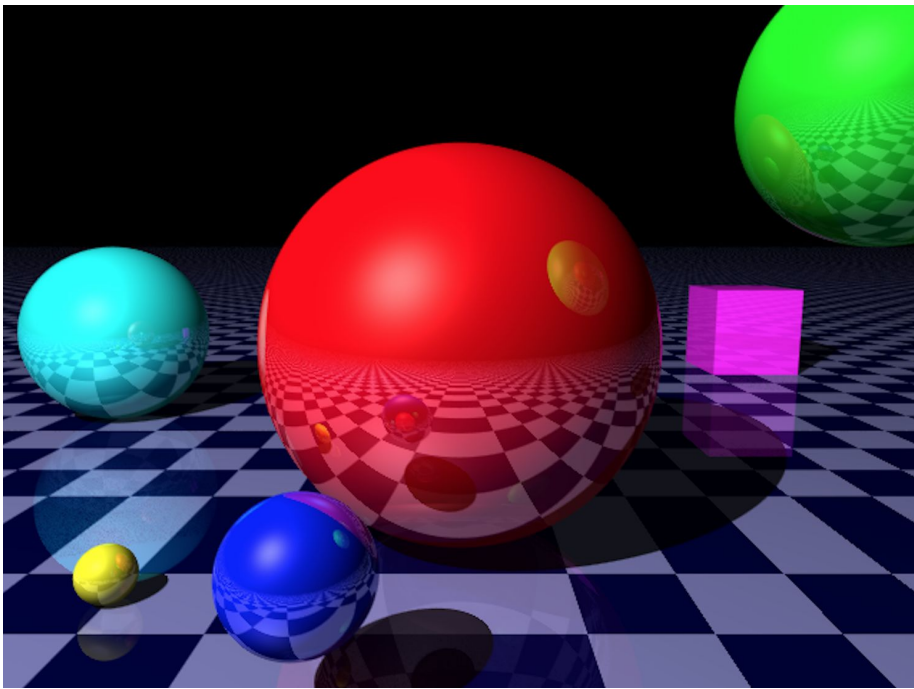
Reflection
(without anti-aliasing)



Shadow
(without anti-aliasing)

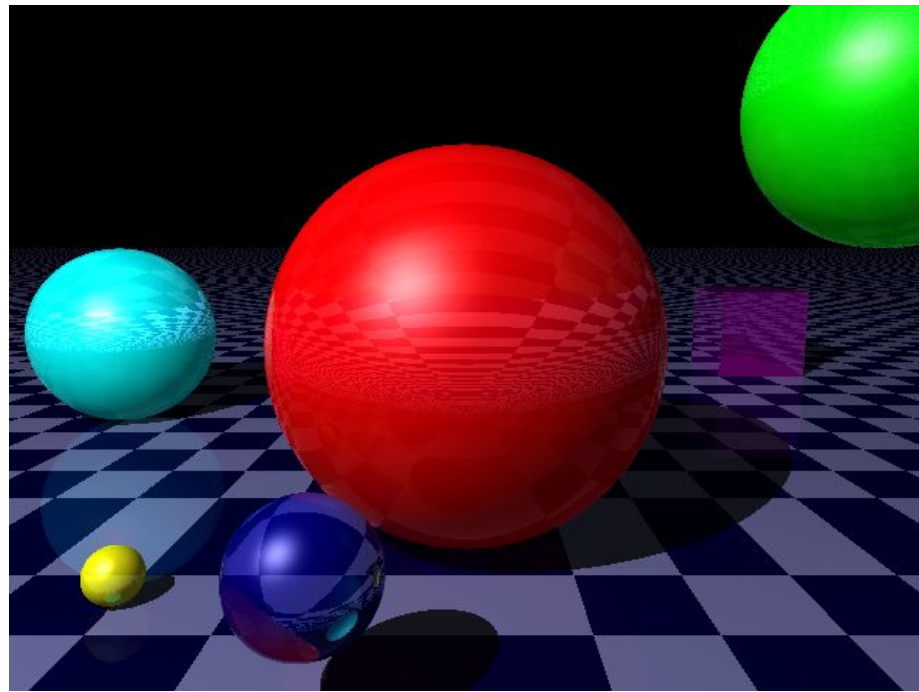


Anti-aliasing



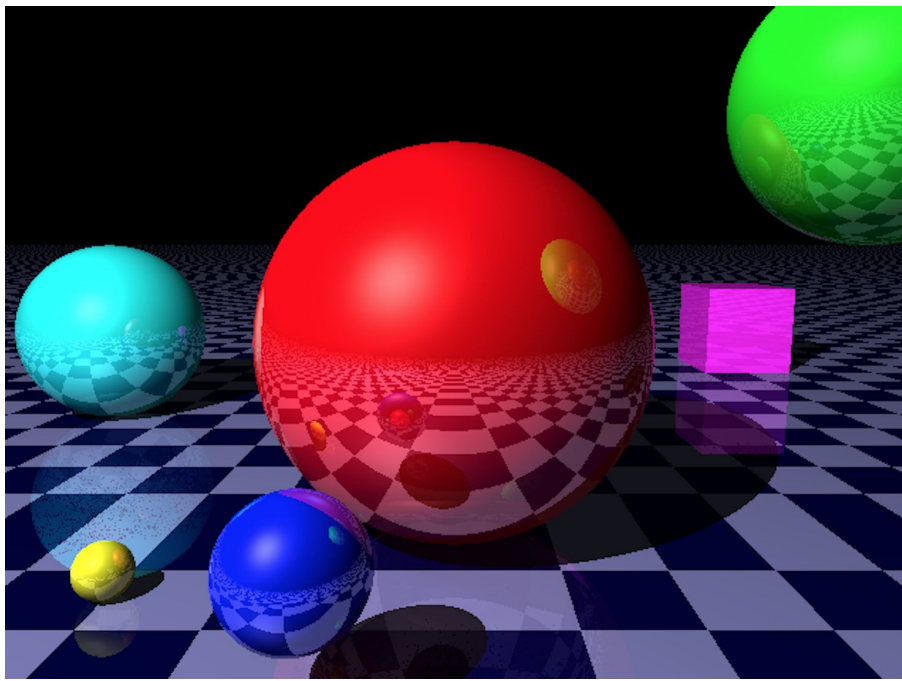
● Reflection + Shadows

● Reflection + Refraction + Shadows



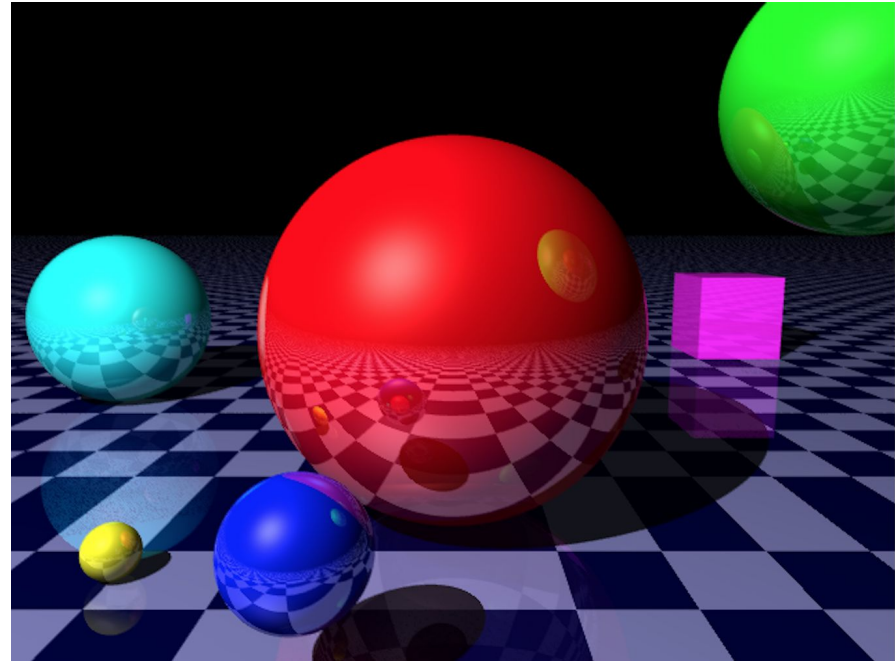
Anti-Aliasing

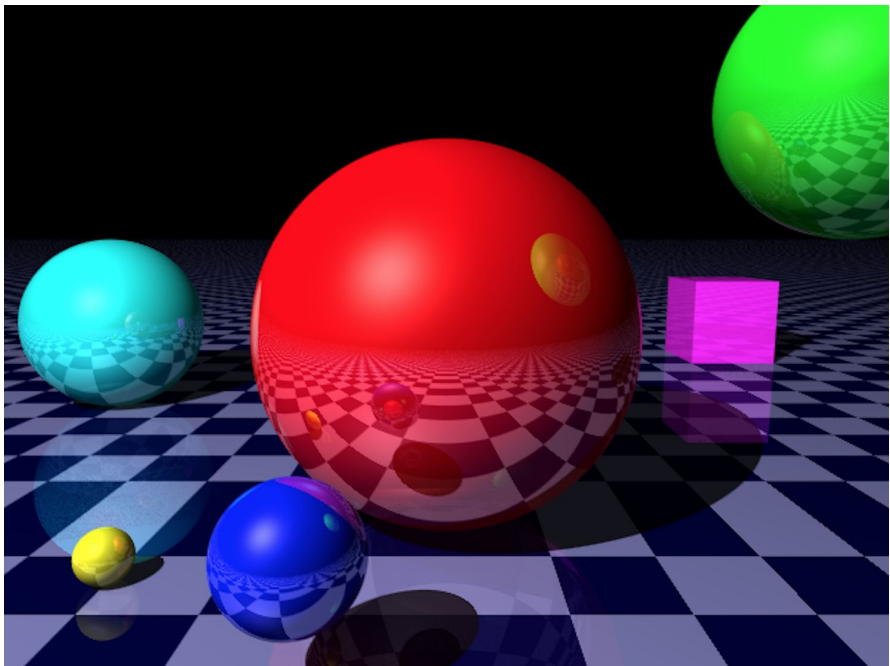
A wireframe horse model is shown against a background of a pixelated orange and red gradient. The horse is rendered with a black wireframe mesh, and its texture is highly pixelated, showing large, distinct squares of color. A white diagonal line runs from the top left to the bottom right, passing over the horse and the text. The text "Anti-Aliasing" is written in a large, white, sans-serif font, centered over the horse's body.



● No Anti-Aliasing

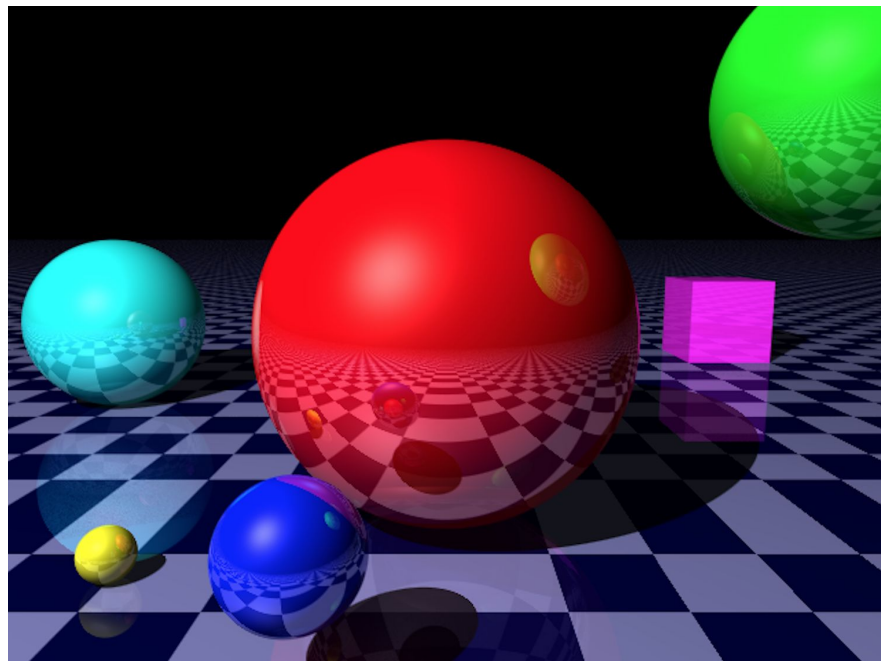
Anti-Aliasing Depth = 5 ●





● Anti-Aliasing Depth = 10

Anti-Aliasing Depth = 20 ●



Technical Challenges

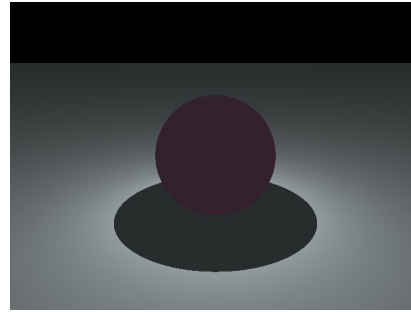
01 | Ray-Object Intersection

02 | Shadow Implementation

03 | Shadow Acne

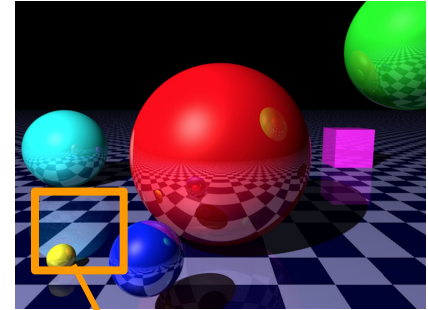
04 | Refraction

02

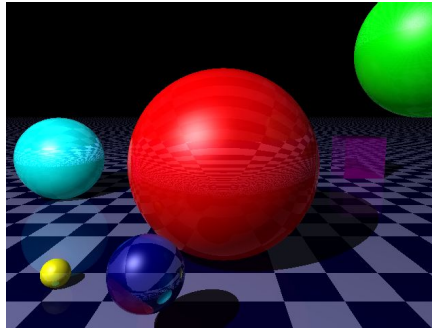


Buggy Shadow Implementation

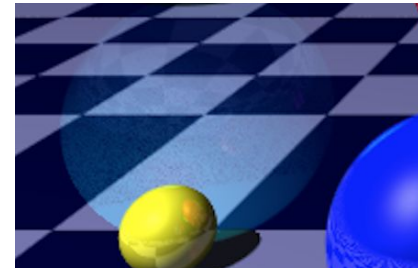
03



04



Unrealistic Refraction



Shadow Acne

Future Work

- 01 | Removing Shadow Acnes
- 02 | Implement transparency
- 03 | Add an Interactive component





Thank you!

