Ray Tracing

Peter Tsun and Brian Mueller

Part 1: Theory

Peter Tsun

Background: Ray Tracing

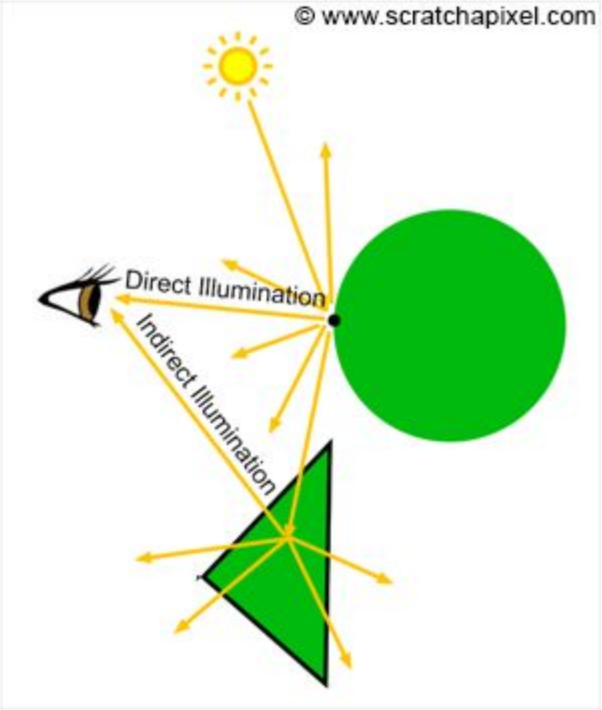
In 1968, Artur Appel came up with a way to render a scene on a screen by sending rays through each pixel on the screen to determine the color for the pixel.

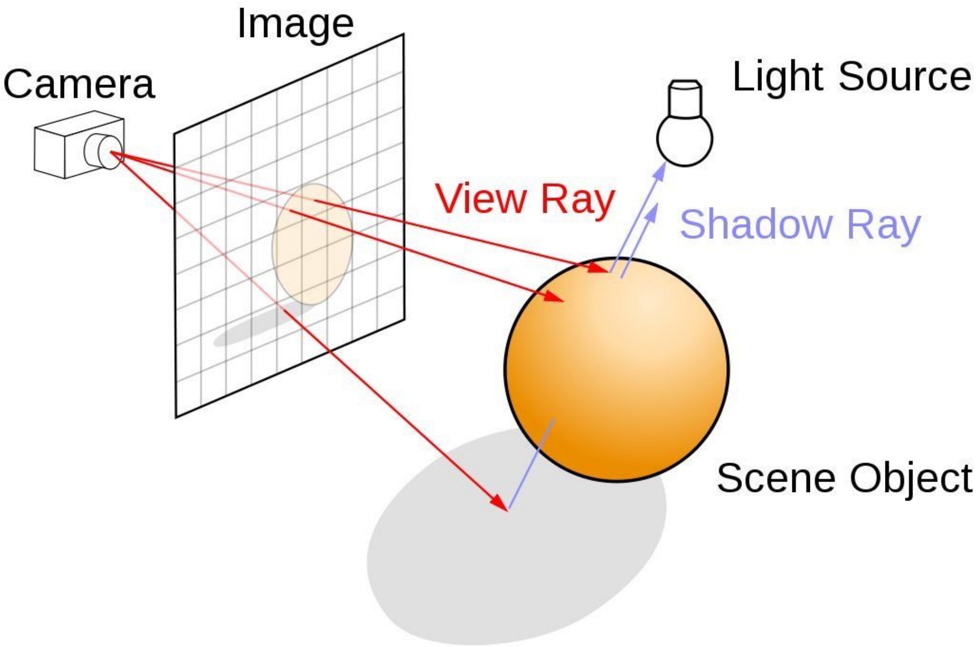
Cars 2006

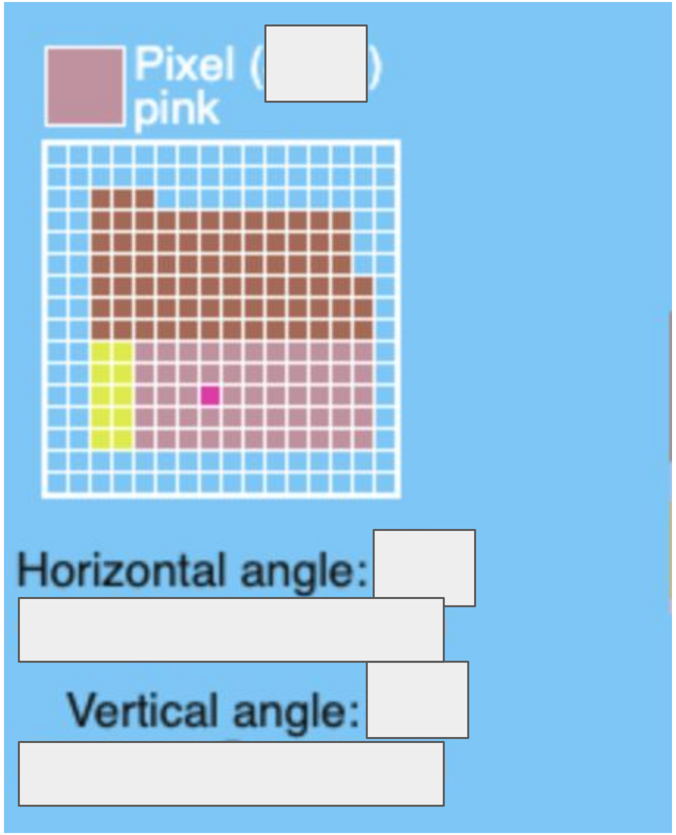
Ray Tracing

Planes 2013

Light-based Ray Tracing in Nature Eye-based Ray Tracing in Image Rendering







On Slack:

Activity

What pixel, horizontal angle, and vertical angle render a PINK pixel?

Pixel: #,#; H: #; V: #

Example:

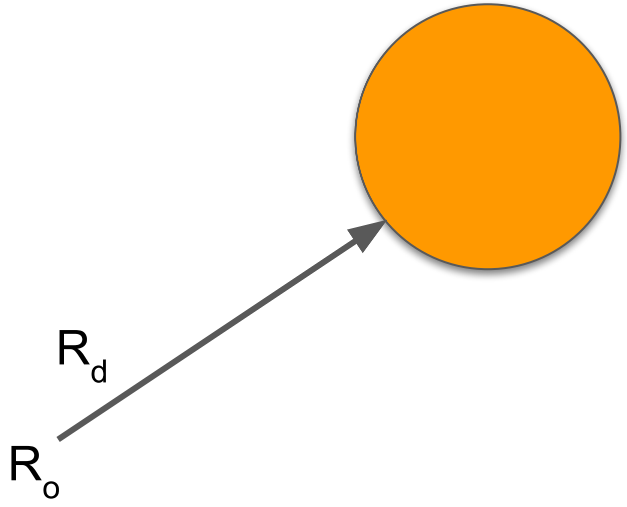
- *Pixel: 10,12; H: 30; V: -82*

Note: there is more than one right answer!

A 3-minute breakout session

In your group, discuss what you know about vectors and discriminant in a quadratic formula. Prepare to share your thoughts with the large group.

Ray-Sphere Intersection



RAY

$R_o = [\ X_o\ Y_o\ Z_o\ ]$ (Ray Origin)

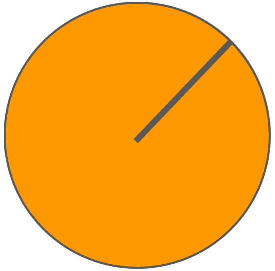
$R_d = [\ X_d\ Y_d\ Z_d\ ]$ (Ray Direction)

$X_d^2 + Y_d^2 + Z_d^2 = 1$ (R_d normalized)

Parametric Form

$R(t) = R_o + R_d\, t$, where $t > 0$

An Implicit Form Describing a Sphere



$S_C = [\ X_C\ \ Y_C\ \ Z_C\ ]$ (Sphere's Center)

S_r (Sphere's Radius)

$[\ X_S\ \ Y_S\ \ Z_S\ ]$ (a point on surface of sphere)

$$(X_S - X_C)^2 + (Y_S - Y_C)^2 + (Z_S - Z_C)^2 = S_r^2$$

$R(t) = R_o + R_d\, t$, where $t > 0$

Values of A, B, C are for finding t (length of a ray)

$[X(t)\ Y(t)\ Z(t)] = [\ X_o\ Y_o\ Z_o\ ] + [\ X_d\ Y_d\ Z_d\ ]\ t$, or

$X(t) = X_o + X_d\, t$, $Y(t) = Y_o + Y_d\, t$, $Z(t) = Z_o + Z_d\, t$

$X(t) = X_S$ and $Y(t) = Y_S$ and $Z(t) = Z_S$

$(X_S - X_C)^2 + (Y_S - Y_C)^2 + (Z_S - Z_C)^2 = S_r^{\,2} \rightarrow At^2 + Bt + C = 0$,

$A = X_d^{\,2} + Y_d^{\,2} + Z_d^{\,2}$, $C = (X_o - X_c)^2 + (Y_o - Y_c)^2 + (Z_o - Z_c)^2 - S_r^{\,2}$

$B = 2[(X \quad X)X + (Y \quad Y)Y + (Z \quad Z)Z]$

$At^2 + Bt + C = 0$

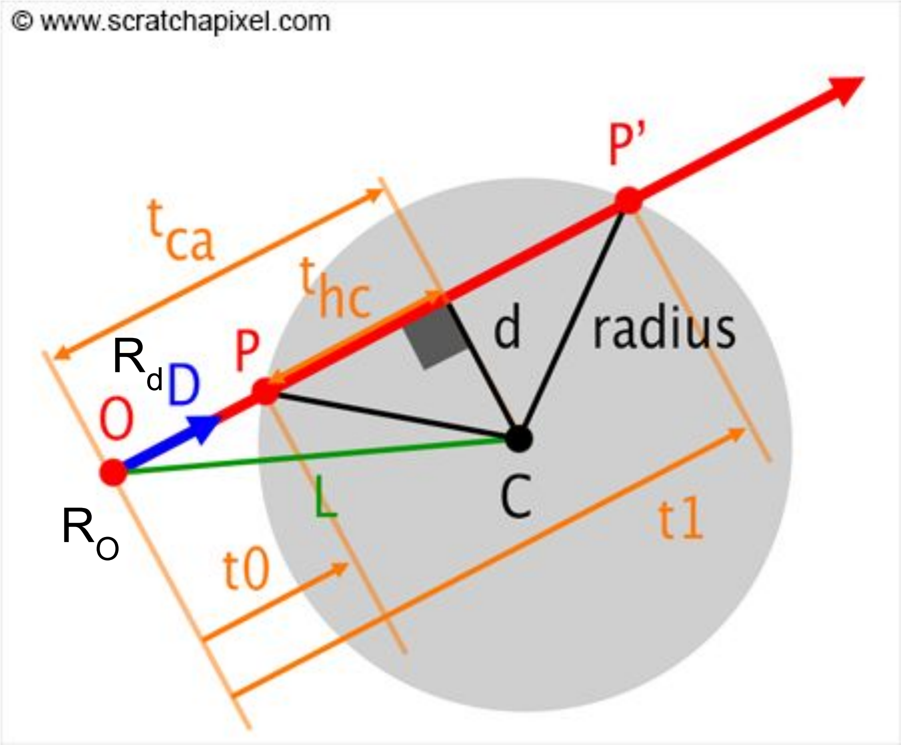
$t = \frac{-B \pm \sqrt{B^2 - 4AC}}{2a}$

$B^2 - 2AC < 0$ → t = imaginary, no intersection

$B^2 - 2AC = 0$ → t has one solution, 1 intersection

$B^2 - 2AC > 0$ → t has two solutions, 2 intersections

Geometric Approach



$L = C - R_O$

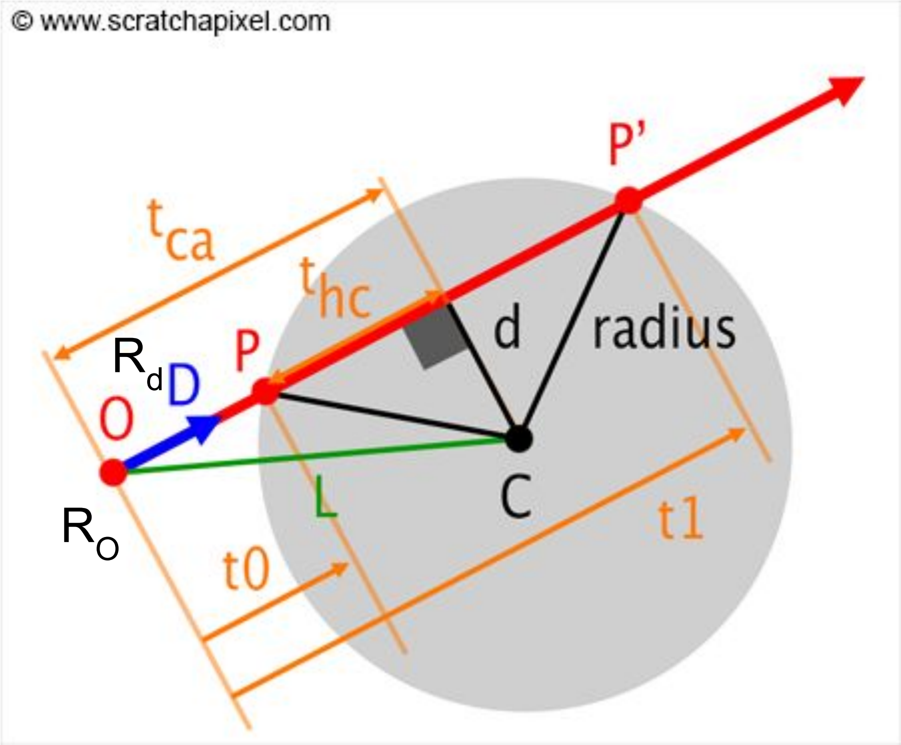
$t_{ca} = L \cdot R_d = L \cos(\theta)$
(Dot Product of L & R_d)

$t_{ca} < 0$ means no intersection

$d^2 = L \cdot L - t_{ca}^2$, r = radius

$d > r$ means no intersection

Geometric Approach



$L = C - R_O$

$t_{hc} = \text{sqrt}\ (\ r^2 - d^2\ )$

$t = t_{ca} - t_{hc}$ or $t = t_{ca} + t_{hc}$

Choose the smaller t

What is your take on mathematics?

Do you feel positive about math?

Do you have negative feeling towards mathematics?

Share your feeling on the slack poll with 1 being negative, 2 being neutral, and 3 being positive

Part 2: Coding

Brian Mueller

Ray Tracing + Code *(for beginners)*

- C++
- Java
- Python
- Javascript
- etc.

C++

PROs

- Fast
- Good resources
  - Section 5 (quadratic formula)

CONs

- Very difficult for beginners
- We have no prior knowledge

Conclusion

👎

Java

PROs

- Great for Minecraft
- Ties to Kotlin (we'll see on the next slide)

CONs

- Difficult to setup quickly

Conclusion

👎

Python

PROs

- Good resources
  - One | [demo]
    - Animation with Kotlin
  - Two | [demo]
- Easy to set up
- Easy to modify

CONs

- Still slow

Conclusion

👎

Javascript

PROs

- Good resources
- Easy to set up
- Easy to modify
- Simple stuff is fast

CONs

- We didn't exclusively learn JS
    - Similar to Java
    - We've done P5js
- Not great for advanced graphics

Conclusion

👍

Javascript Ray Tracer

- Out of the box (example 3)
- Documentation
  - README
  - src/Material.js, Render*, Scene.js → helper files
  - src/Vector3.js and Sphere.js → much of what Peter showed you
  - src/RayTracer.js#L124 → what color is this pixel?
- You wouldn't often code your own ray tracer.
  - It's more important that you're able to use someone else's.
  - The limitation is often their documentation.
  - This one is pretty good, plus I added some HTML/JS to make it even easier to understand
- Modified (examples/hunter)

Breakout Rooms

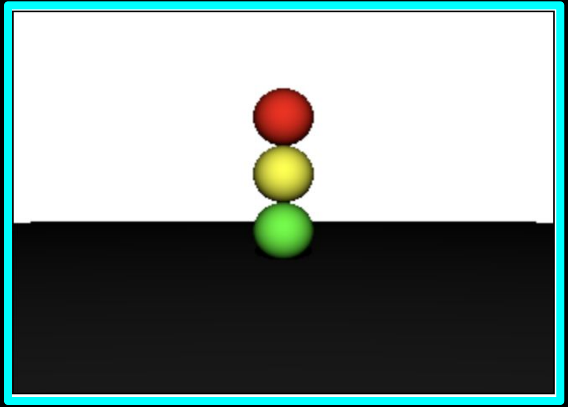
- Fork the repl.
- Preview hunter.html.
- Modify the values in the boxes.
- Open the code for hunter.js
- Look through the code/comments.
- Try commenting out `scene.add(sphere1)` then add your own shape(s).
- Feel free to do the same with light(s).
- Assist each other!

https://replit.com/@briansmueller/raytracer-js

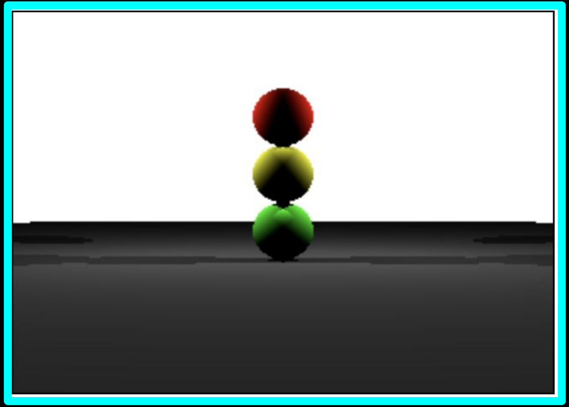
NOTE: you will see comments for the homework. Do not start that yet.

Homework

Task 1: Traffic Signal



Task 2: Two rear lights



Fork the repl, then follow the directions in the comments of hunter.js. Preview hunter.html.

EITHER put the link to your repl in the README of your work repo, OR include the files in your work repo

Async

Find an article about ray tracing and post your thoughts on async slack channel.

Thank you!