Lab #3 - Ray tracing (part 2)

Informática Gráfica

Adolfo Muñoz - Julio Marco
Pablo Luesia - J. Daniel Subías – Óscar Pueyo

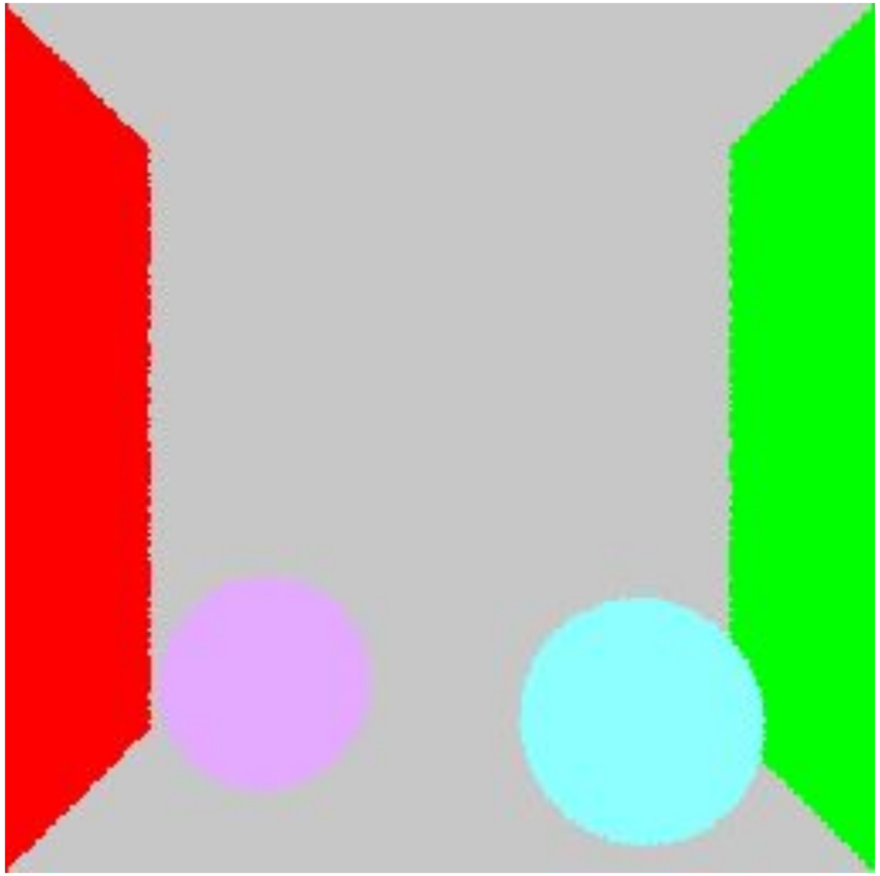
Graphics and Imaging Lab

Before we begin…

- You can generate your first image today :)

Before we begin…

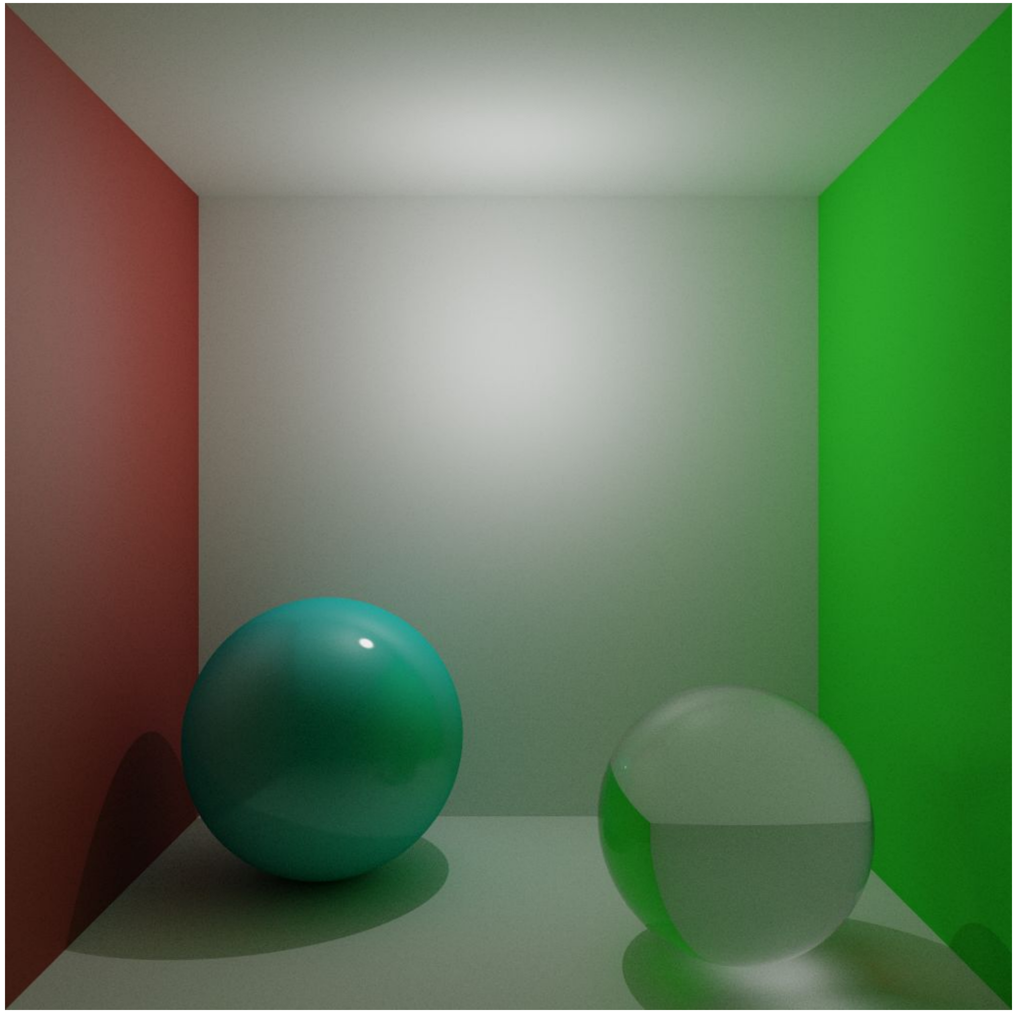
- You can generate your first image today :)



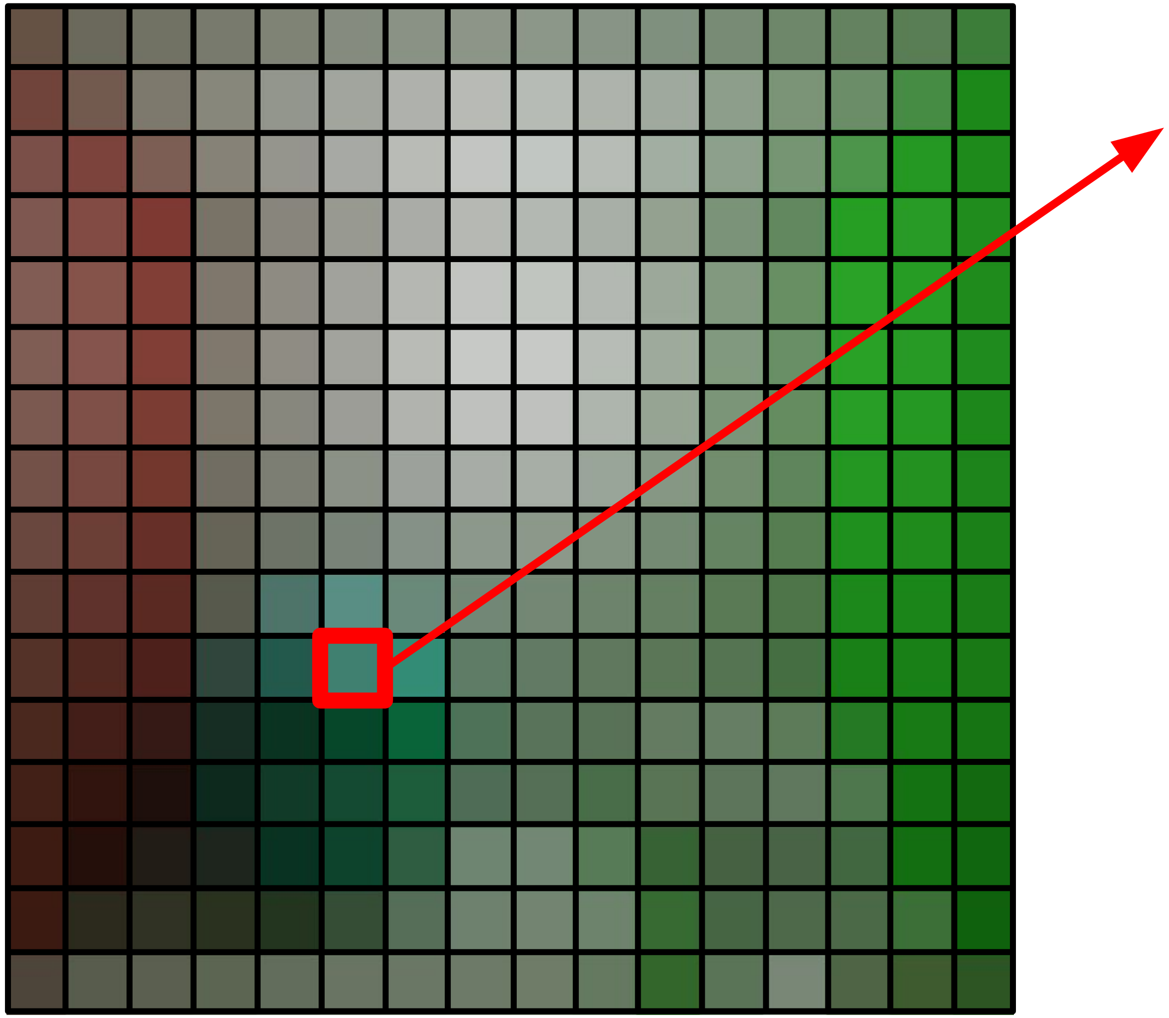
Before we begin…

- You can generate your first image today :)
- Careful with the submissions
  - Lab 3 (ray tracing) does not need to be submitted
    - Recommended deadline: **October 18th**
  - Lab 4 (path tracing) **will be submitted** at the end of the course
    - All of the code you write today will be used for Lab 4
    - Recommended deadline: November 13th (moodle: January 11th, this is only a recommendation)
- Remember: Final work is 80% of the final grade

Which color do we fill each pixel with?



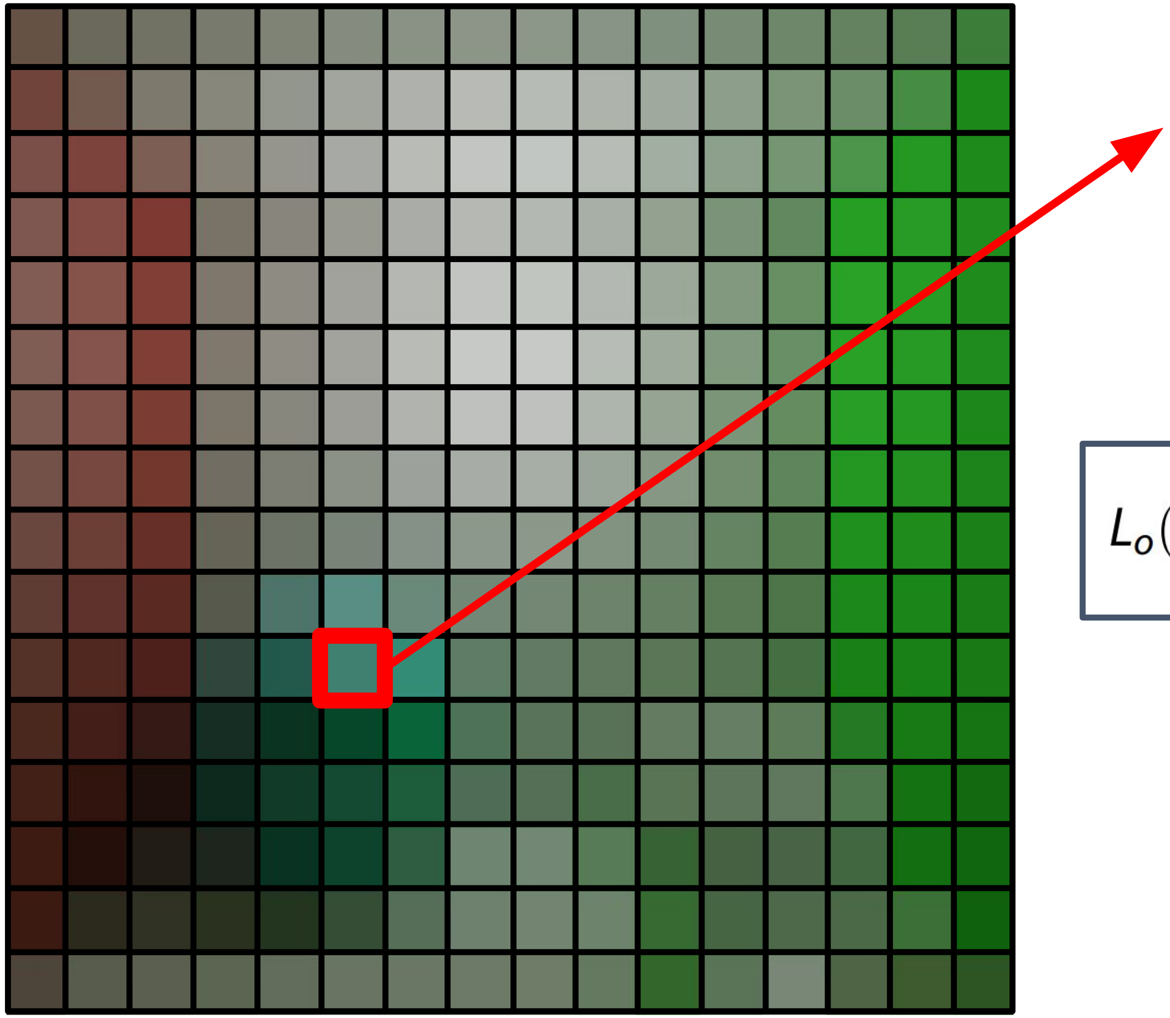
Which color do we fill each pixel with?



R = 0.25 G = 0.5 B = 0.45

But how?

Which color do we fill each pixel with?

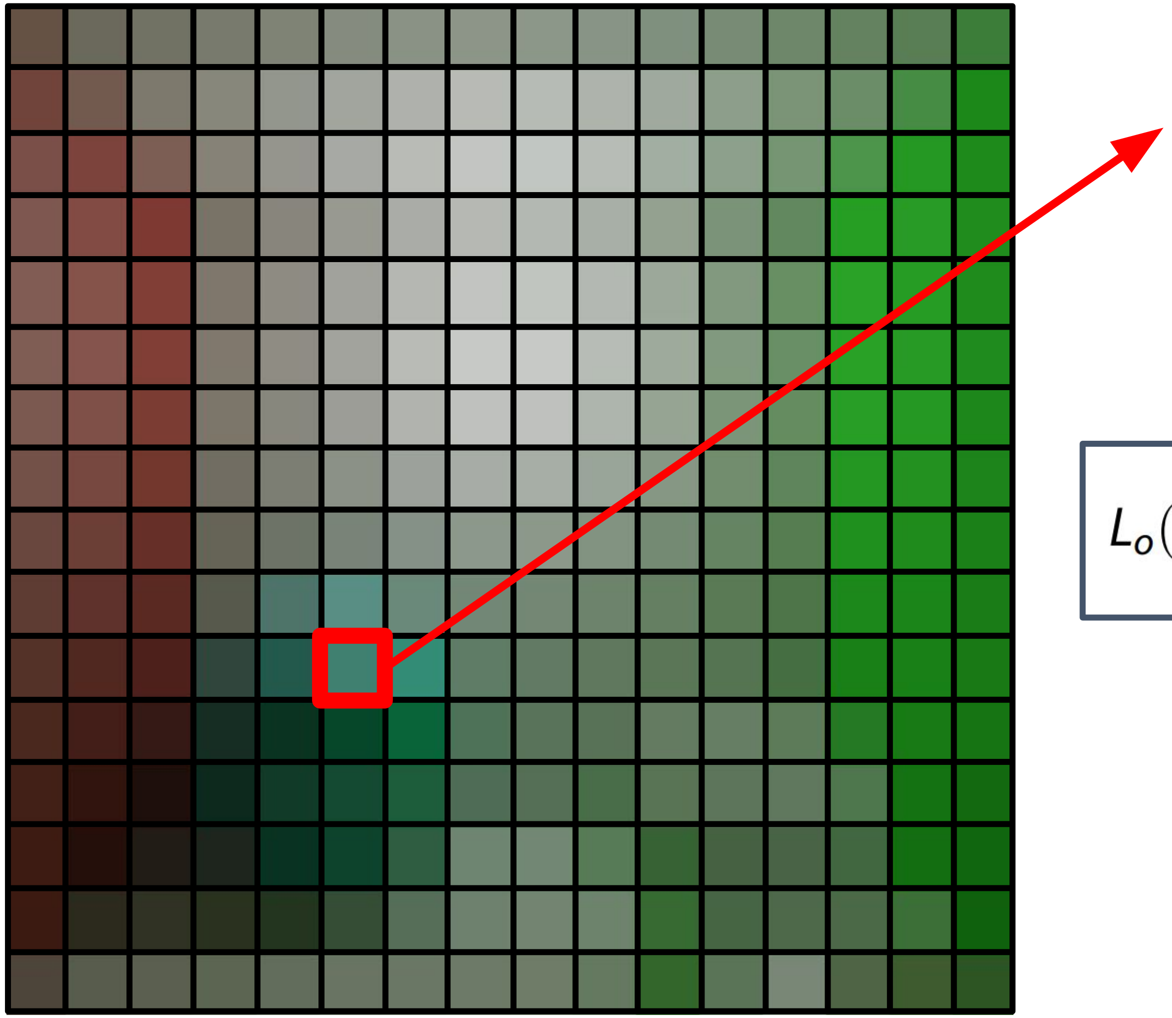


R = 0.25 G = 0.5 B = 0.45

But how?

$$L_o(\mathbf{x},\omega_\mathbf{o}) = L_e(\mathbf{x},\omega_\mathbf{o}) + \int_\Omega L_i(\mathbf{x},\omega_\mathbf{i}) f_r(\mathbf{x},\omega_\mathbf{i},\omega_\mathbf{o}) |\mathbf{n}\cdot\omega_\mathbf{i}| d\omega_\mathbf{i}$$

Which color do we fill each pixel with?



R = 0.25 G = 0.5 B = 0.45

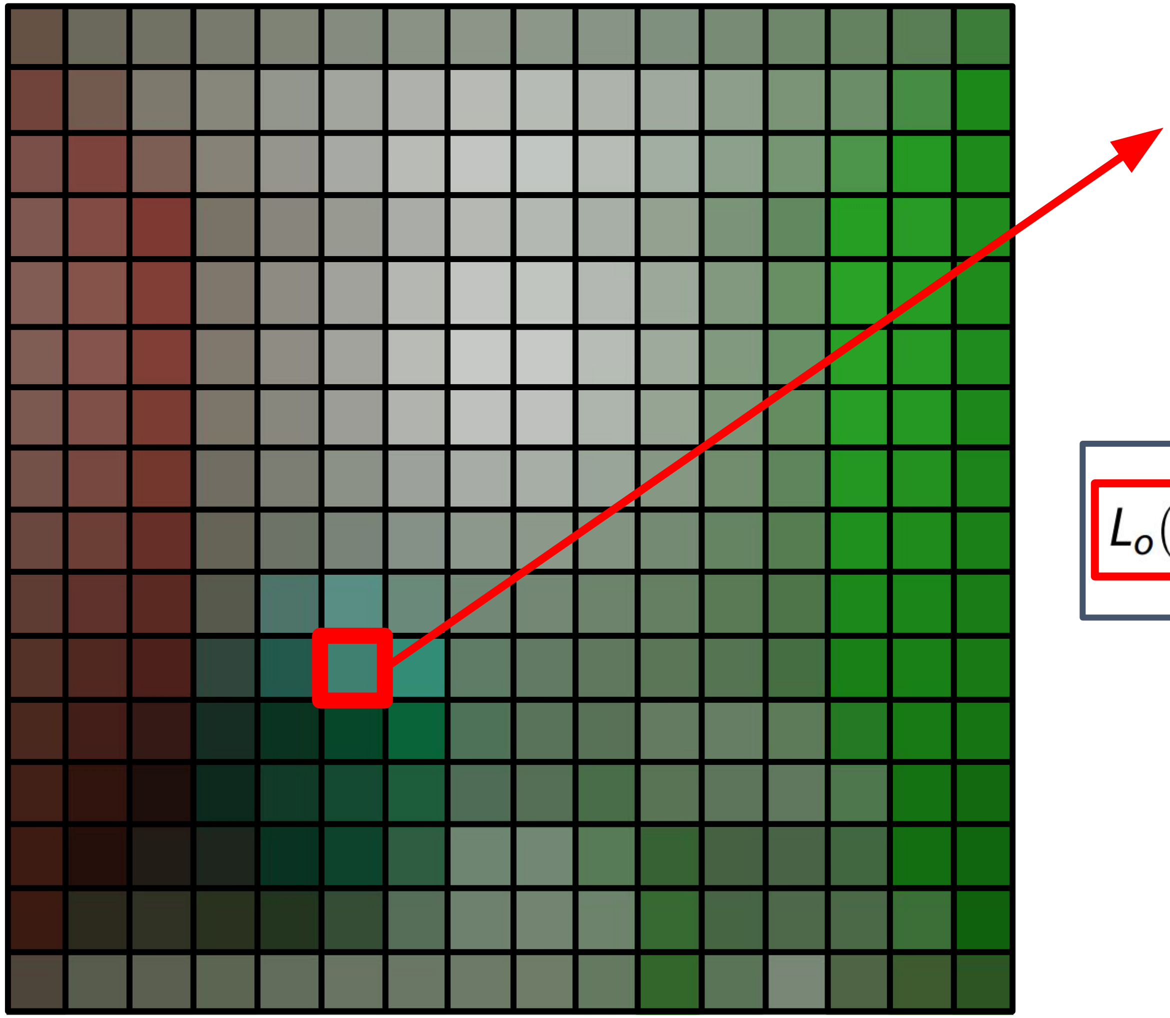
But how?

$$L_o(\mathbf{x},\omega_\mathbf{o}) = L_e(\mathbf{x},\omega_\mathbf{o}) \cancel{+ \int_\Omega L_i(\mathbf{x},\omega_\mathbf{i}) f_r(\mathbf{x},\omega_\mathbf{i},\omega_\mathbf{o}) |\mathbf{n}\cdot\omega_\mathbf{i}| d\omega_\mathbf{i}}$$

In this lab,

we simplify the render equation:

Which color do we fill each pixel with?



R = 0.25 G = 0.5 B = 0.45

But how?

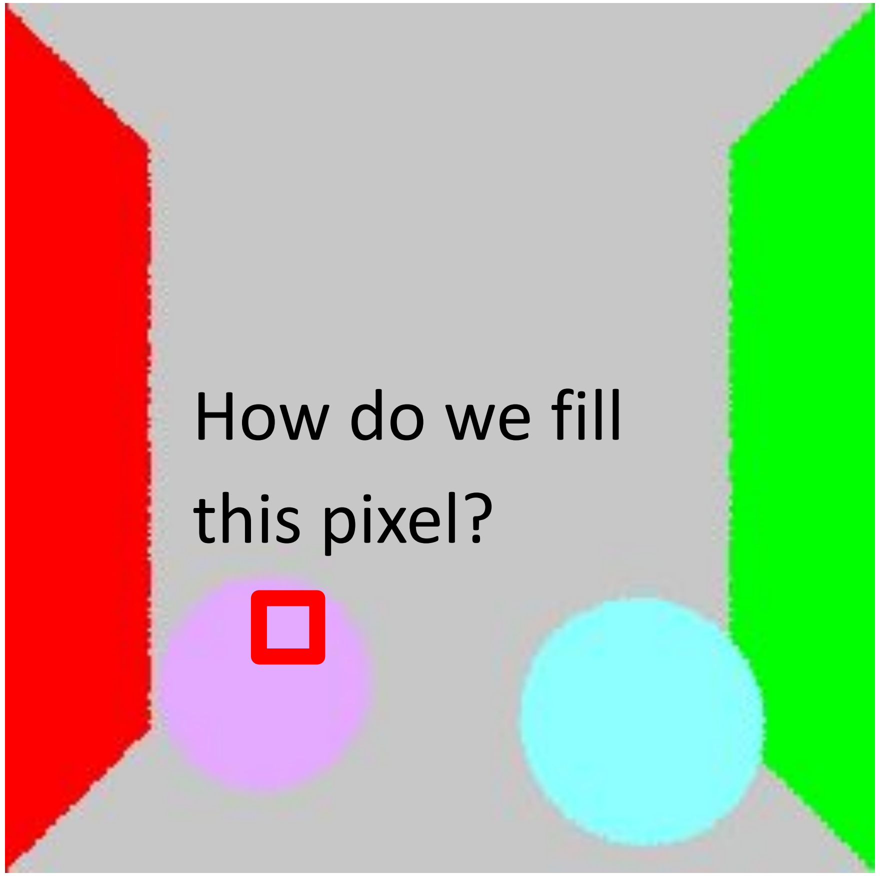
$$\boxed{L_o(\mathbf{x},\omega_\mathbf{o}) = L_e(\mathbf{x},\omega_\mathbf{o})} \cancel{+\int_\Omega L_i(\mathbf{x},\omega_\mathbf{i}) f_r(\mathbf{x},\omega_\mathbf{i},\omega_\mathbf{o})|\mathbf{n}\cdot\omega_\mathbf{i}|\, d\omega_\mathbf{i}}$$

In this lab,

we simplify the render equation:

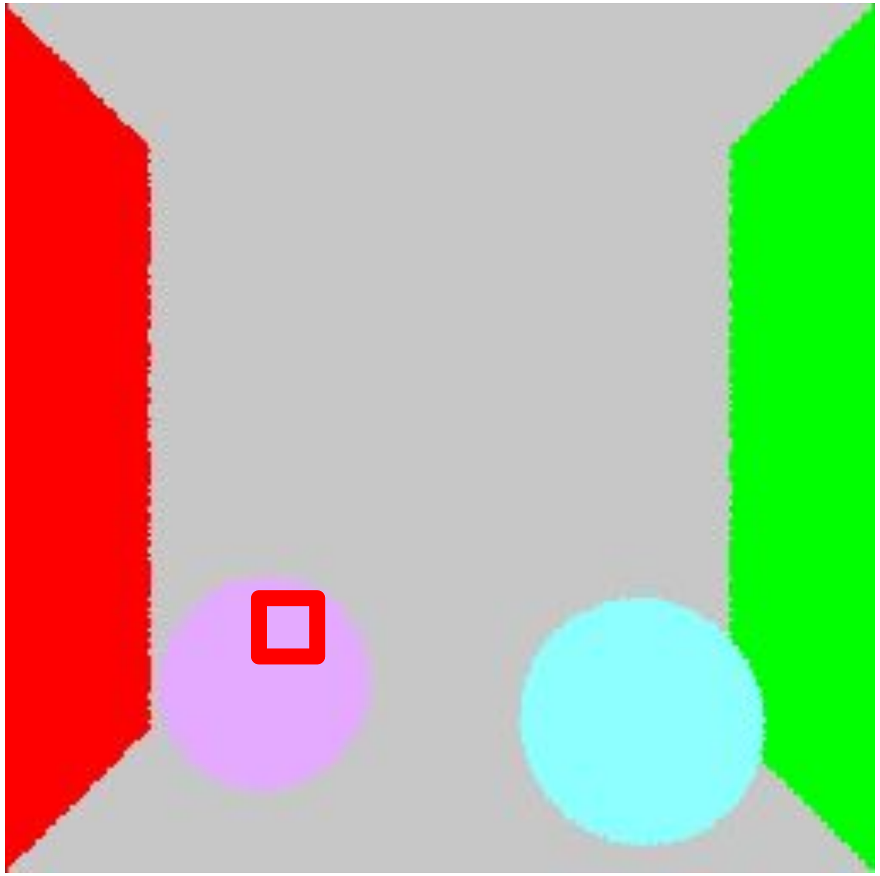
Everything is an area light

Which color do we fill each pixel with?

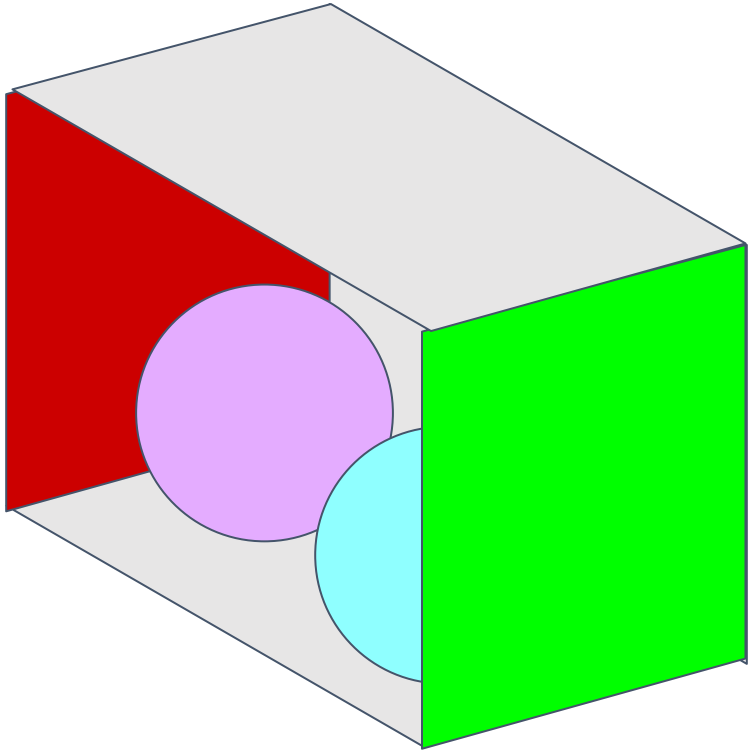


How do we fill this pixel?

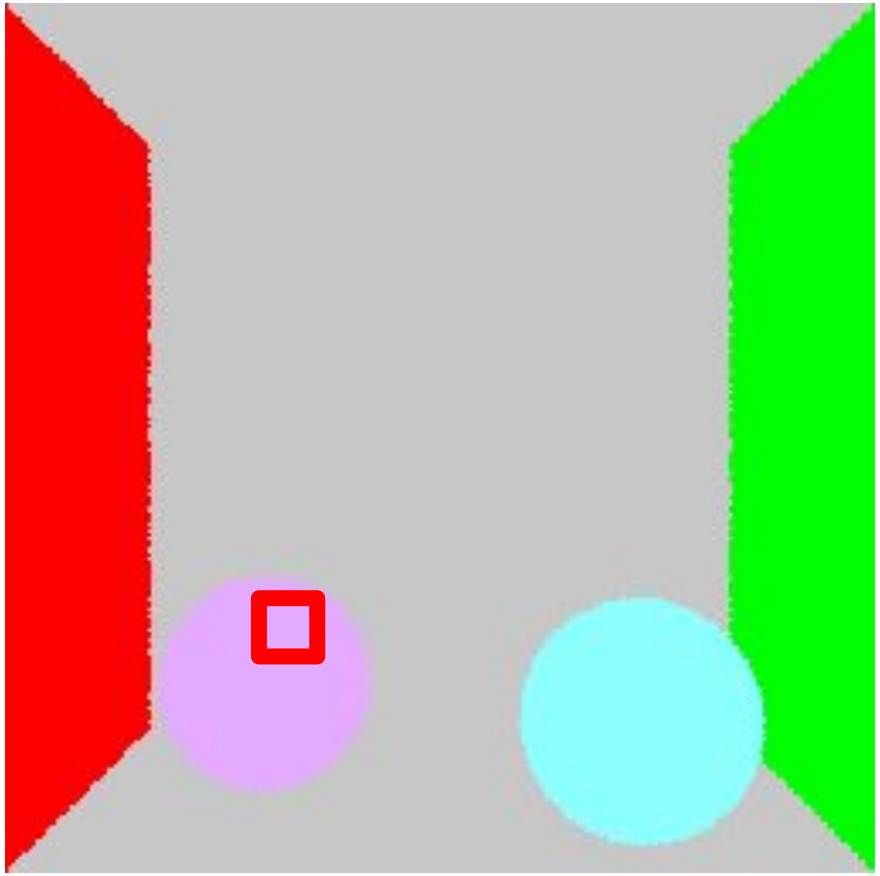
Which color do we fill each pixel with?



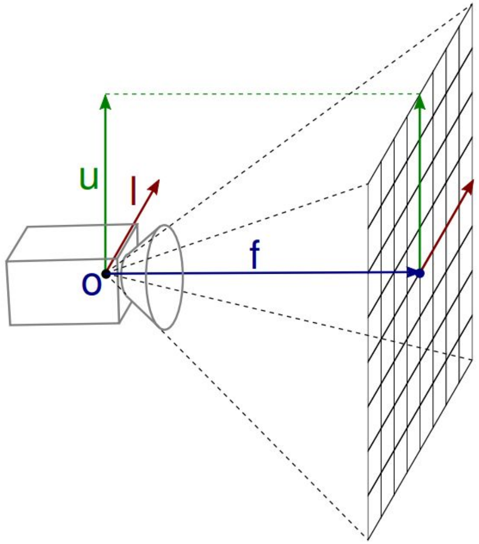
Define your scene's geometry (planes, spheres)

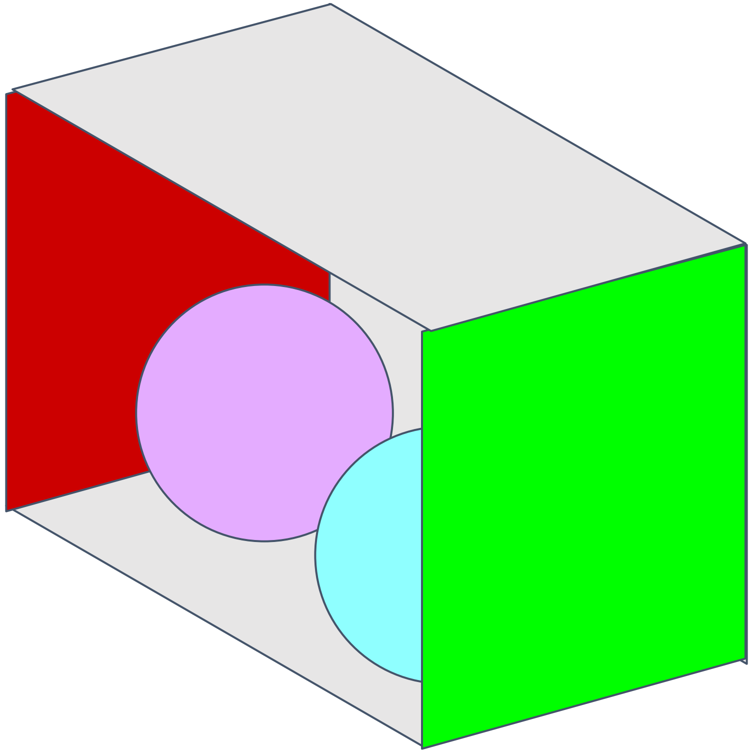


Which color do we fill each pixel with?

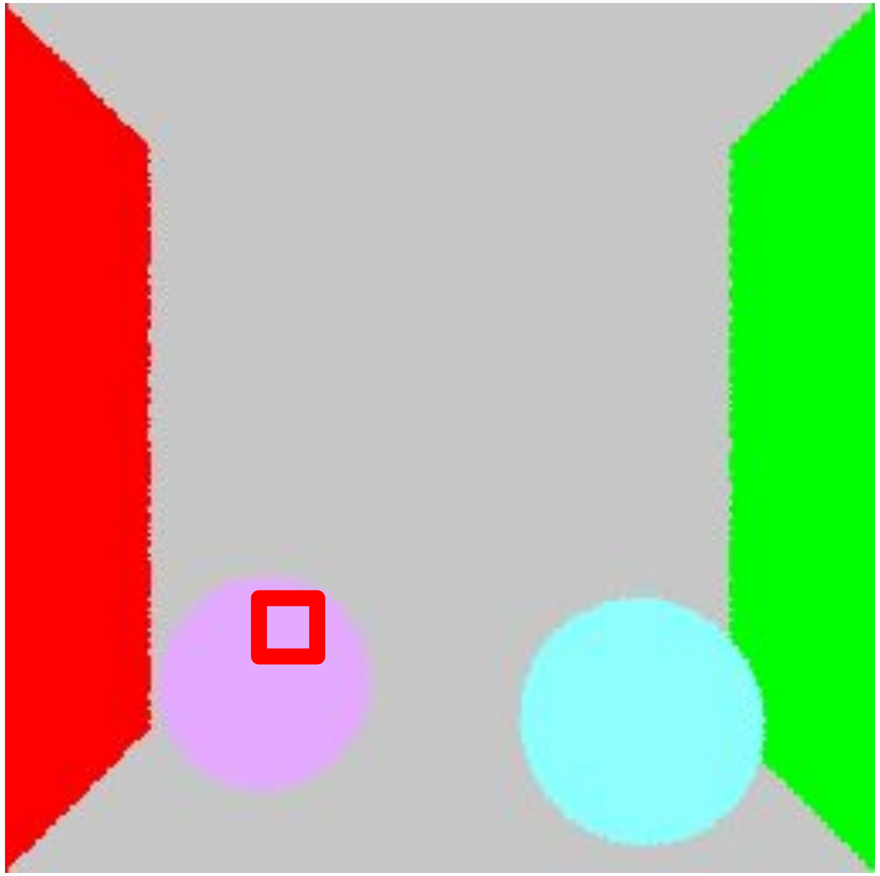


Define your scene's camera (image plane, film)

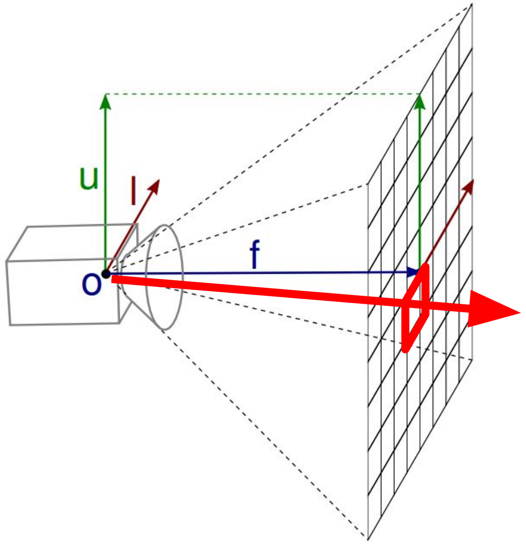


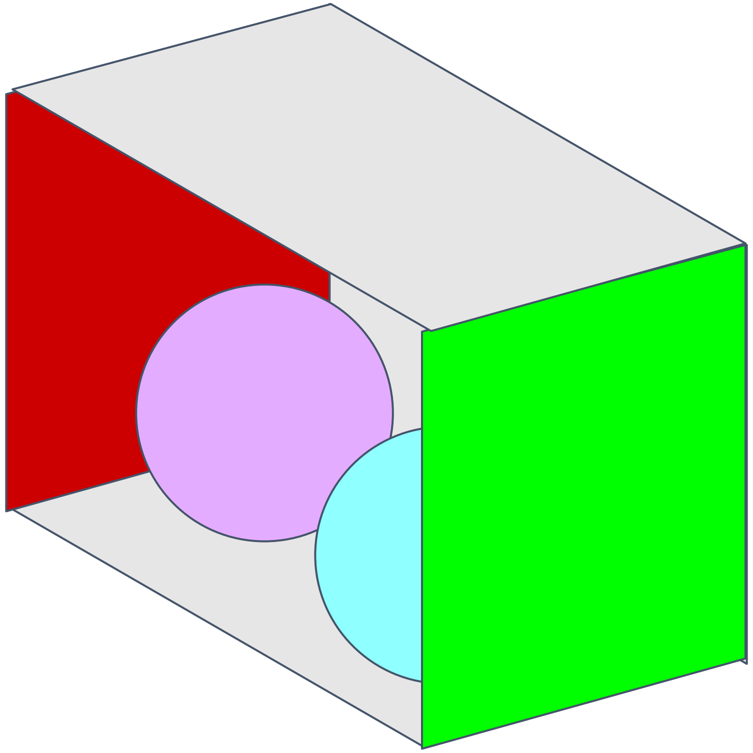


Which color do we fill each pixel with?

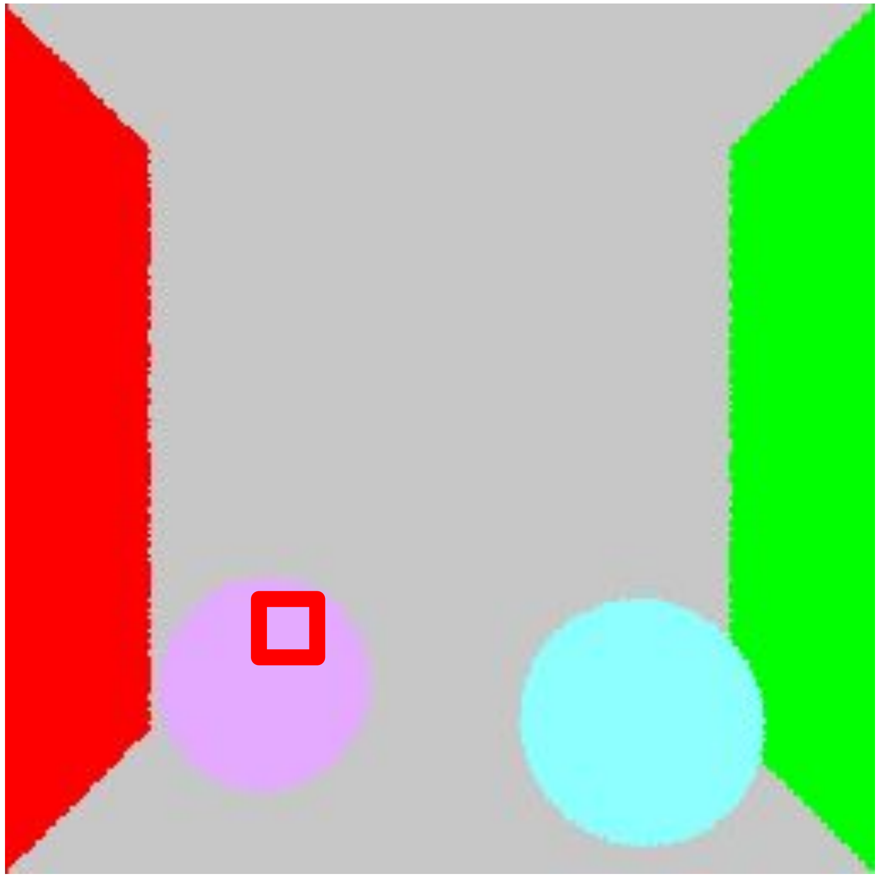


Generate a ray → that passes through □

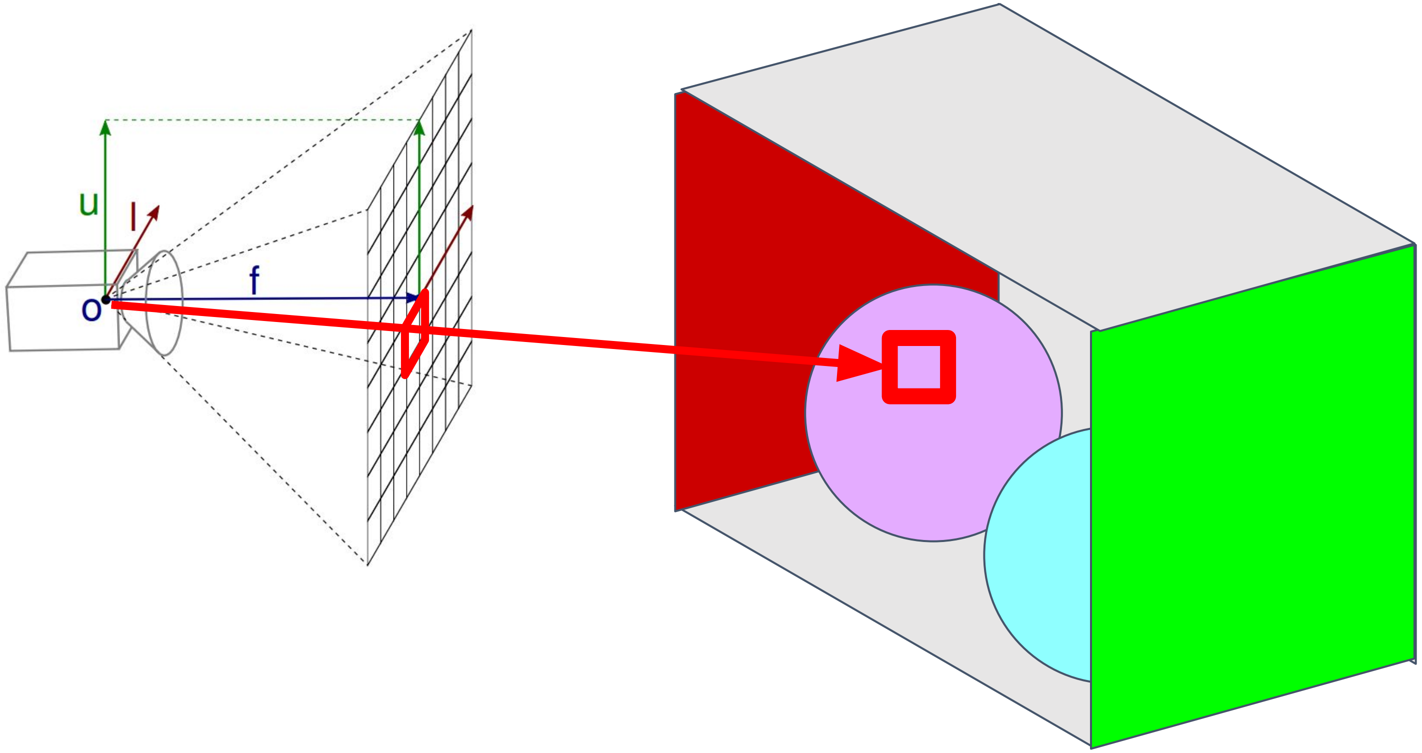




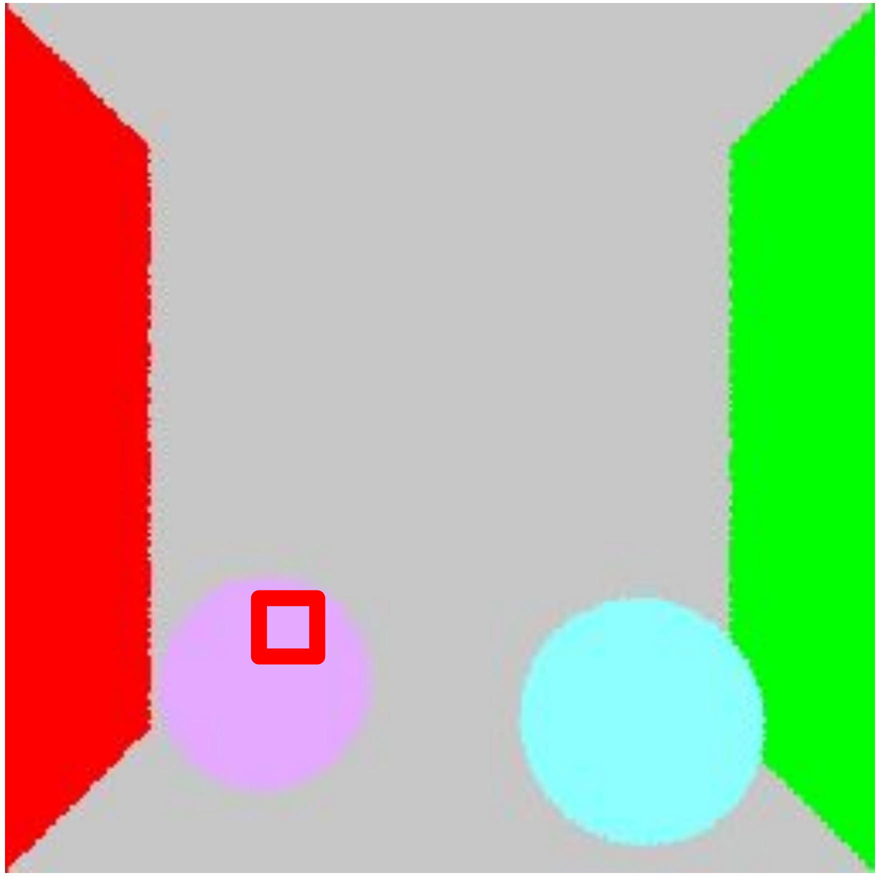
Which color do we fill each pixel with?



Intersect this ray with your scene

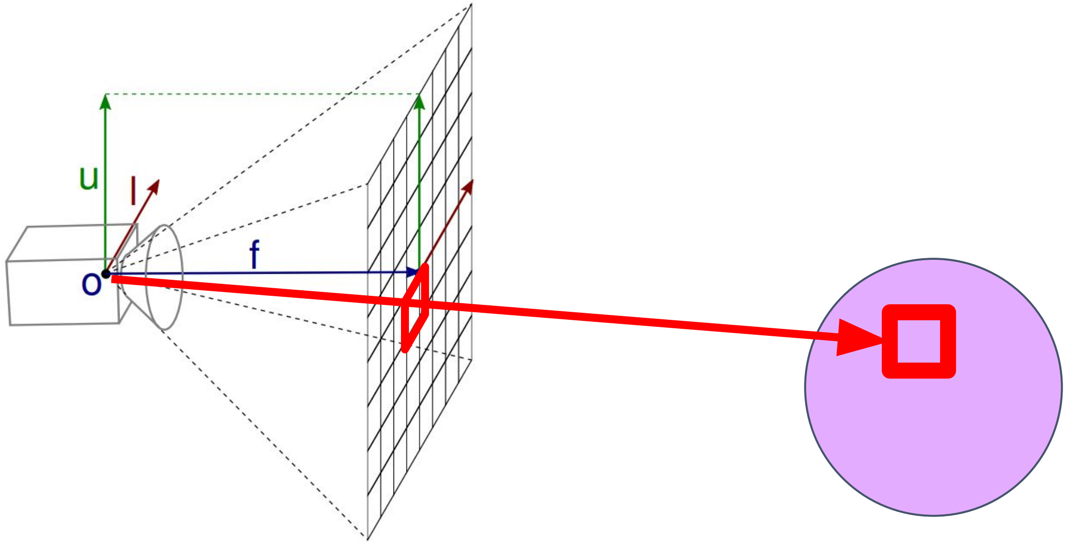


Which color do we fill each pixel with?



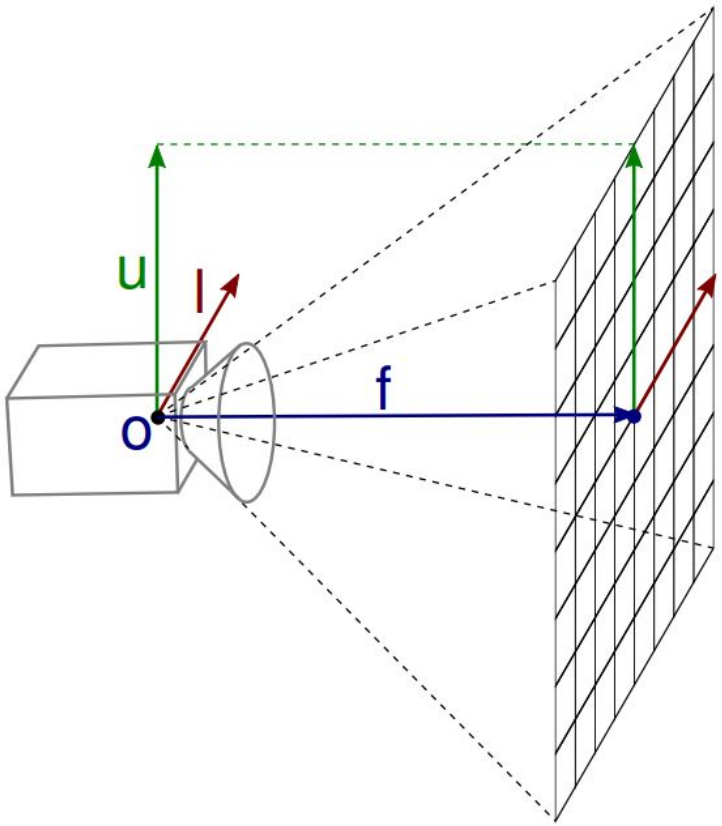
Return your sphere's emission color

(everything is an area light in this lab)



- Center: (x, y, z)
- Radius: d
- Emission: (r, g, b)

Considerations about the camera

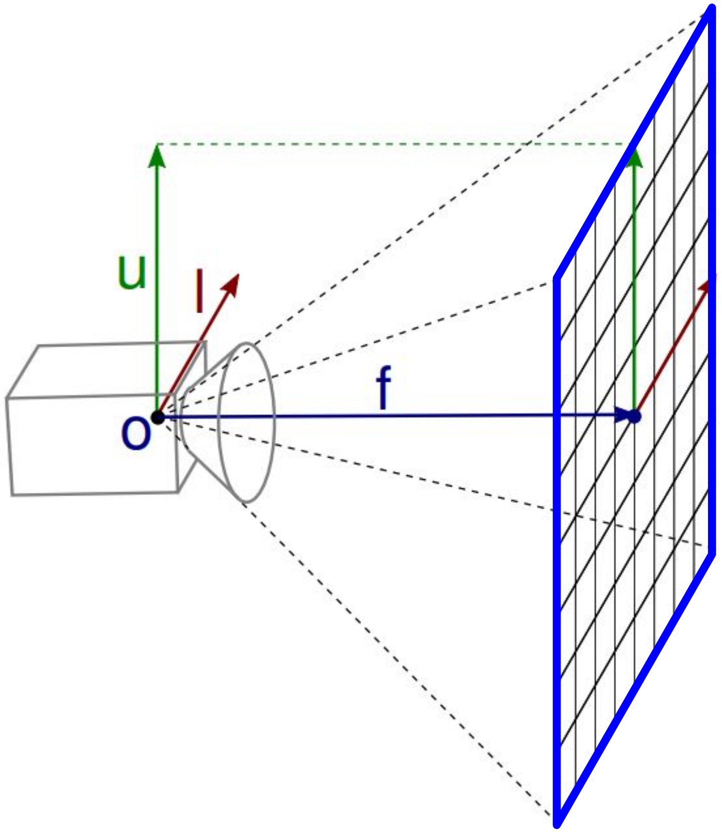


A camera is defined by:

- Local coordinate system

  (left L, up U, forward F, origin O)

Considerations about the camera



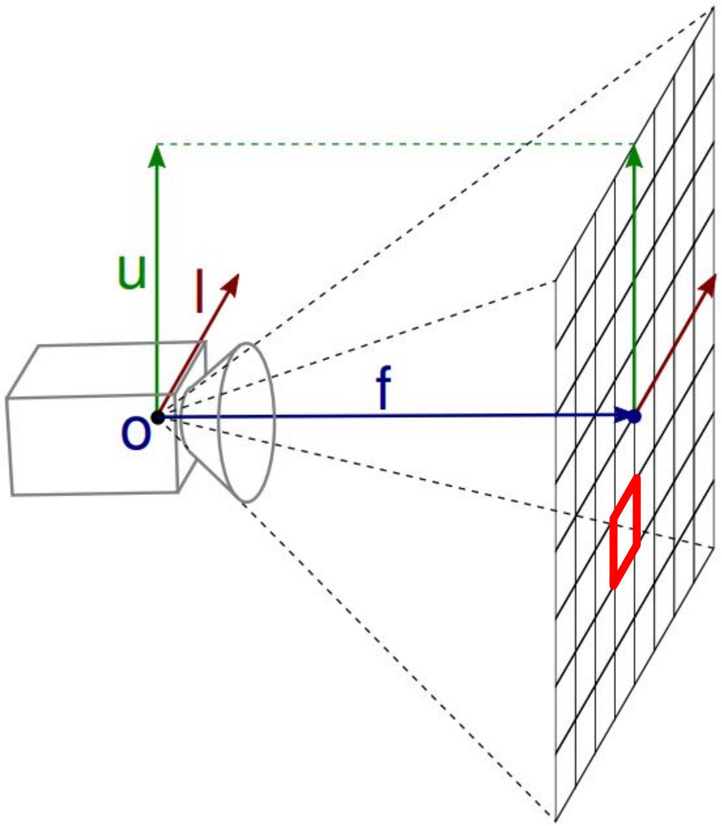
A camera is defined by:

- Local coordinate system

  (left L, up U, forward F, origin O)

- Forward, up & left also define the image plane
  - Divided in pixels (width, height)
  - Pixels should be square (not necessarily)

Considerations about the camera



A camera is defined by:

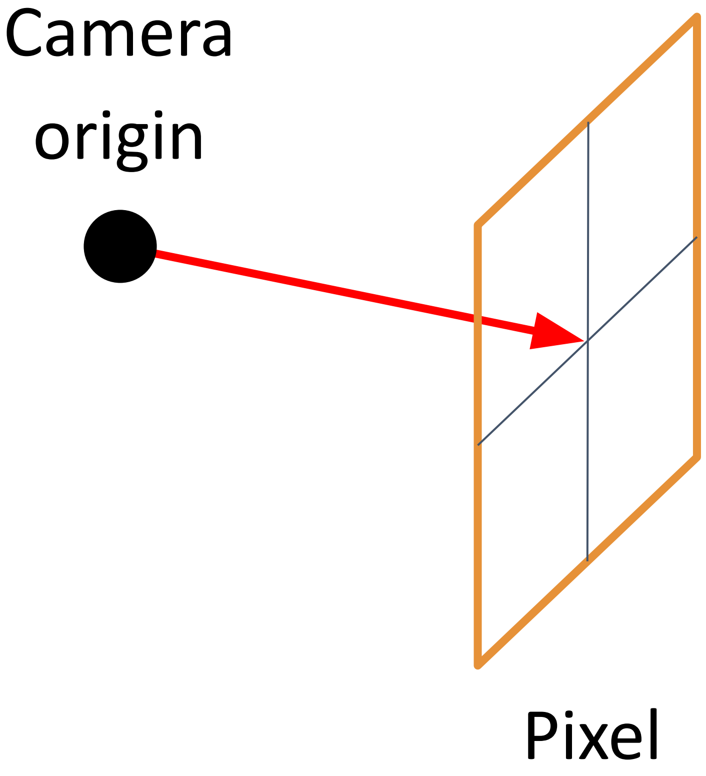
- Local coordinate system

  (left L, up U, forward F, origin O)

- Forward, up & left also define the image plane
  - Divided in pixels (width, height)
  - Pixels should be square (not necessarily)

- Calculate the boundaries of a pixel
  - Rays: origin O, direction towards pixel

Considerations about the camera



A camera is defined by:

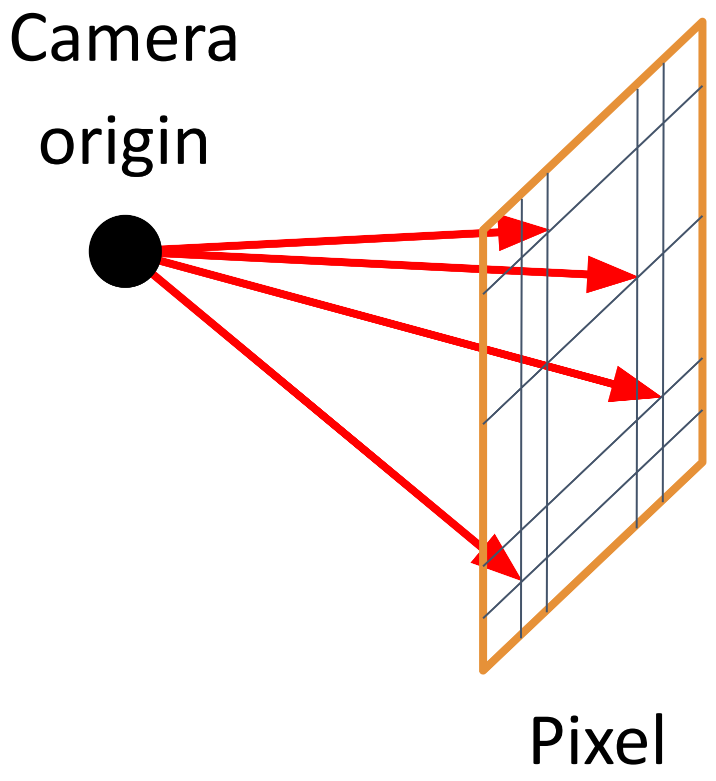
- Local coordinate system

  (left L, up U, forward F, origin O)

- Forward, up & left also define the image plane
  - Divided in pixels (width, height)
  - Pixels should be square (not necessarily)

- Calculate the boundaries of a pixel
  - Rays: origin O, direction towards pixel

Considerations about the camera



A camera is defined by:

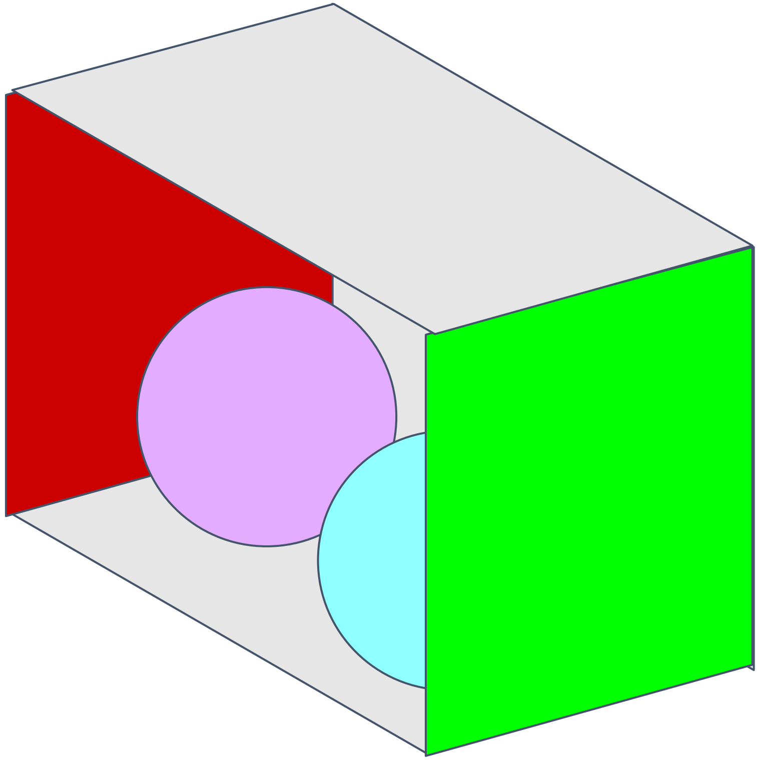
- Local coordinate system

  (left L, up U, forward F, origin O)

- Forward, up & left also define the image plane
  - Divided in pixels (width, height)
  - Pixels should be square (not necessarily)
- Calculate the boundaries of a pixel
  - Rays: origin O, direction towards pixel

Example scene: Cornell Box

- **Geometry**



Planes defined by normal (n) and distance (d)

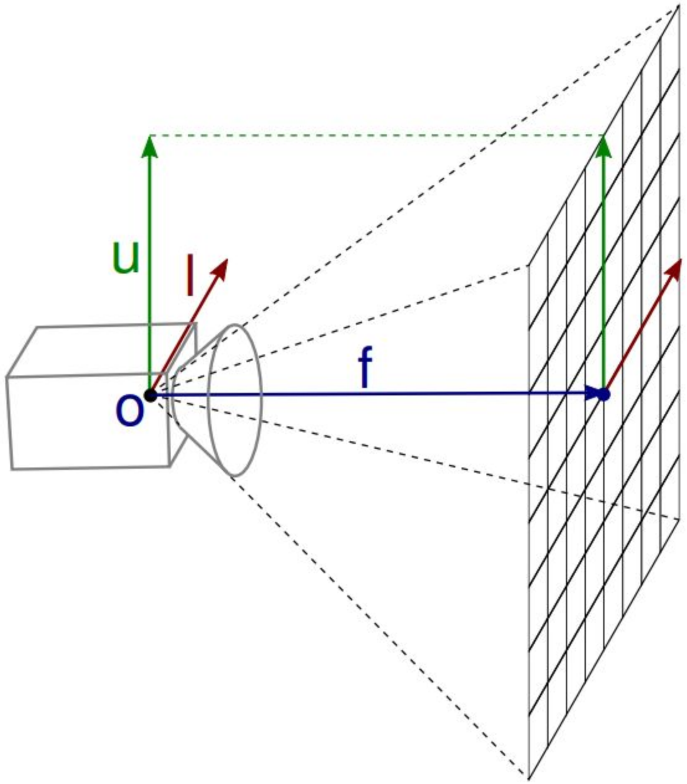
Left plane	n = (1, 0, 0), d = 1
Right plane	n = (-1, 0, 0), d = 1
Floor plane	n = (0, 1, 0), d = 1
Ceiling plane	n = (0, -1, 0), d = 1
Back plane	n = (0, 0, -1), d = 1

Spheres defined by center (c) and radius (r)

Left sphere	c = (-0.5, -0.7, 0.25), r = 0.3
Right sphere	c = (0.5, -0.7, -0.25), r = 0.3

Example scene: Cornell Box

- **Camera**

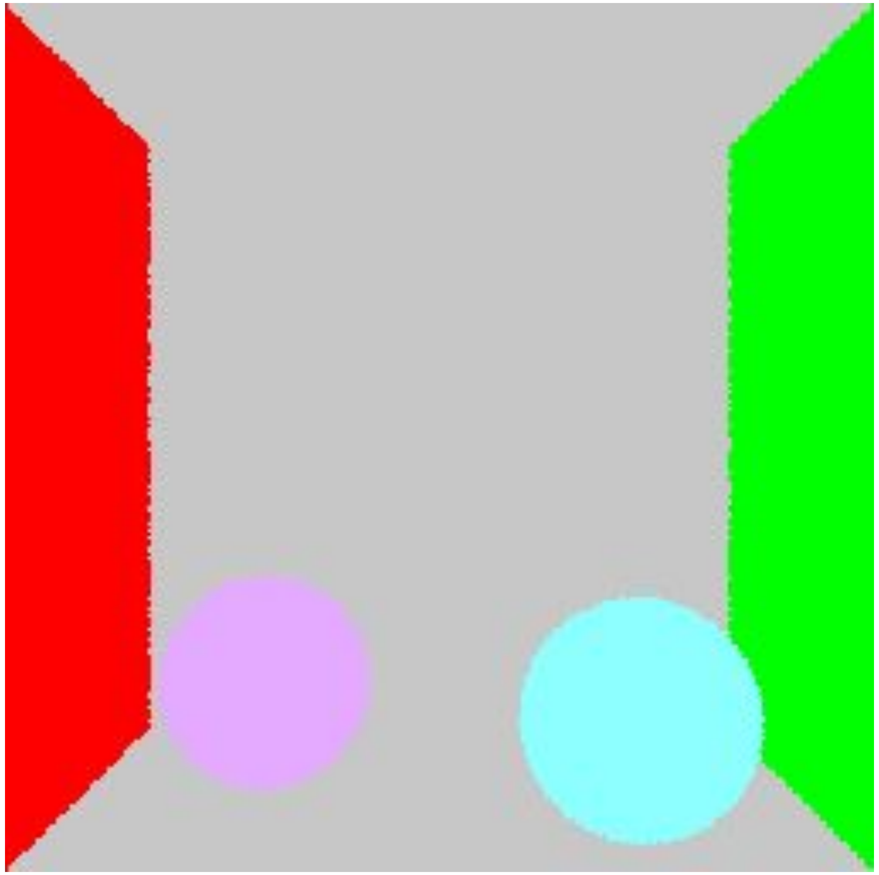


Camera and image plane defined by

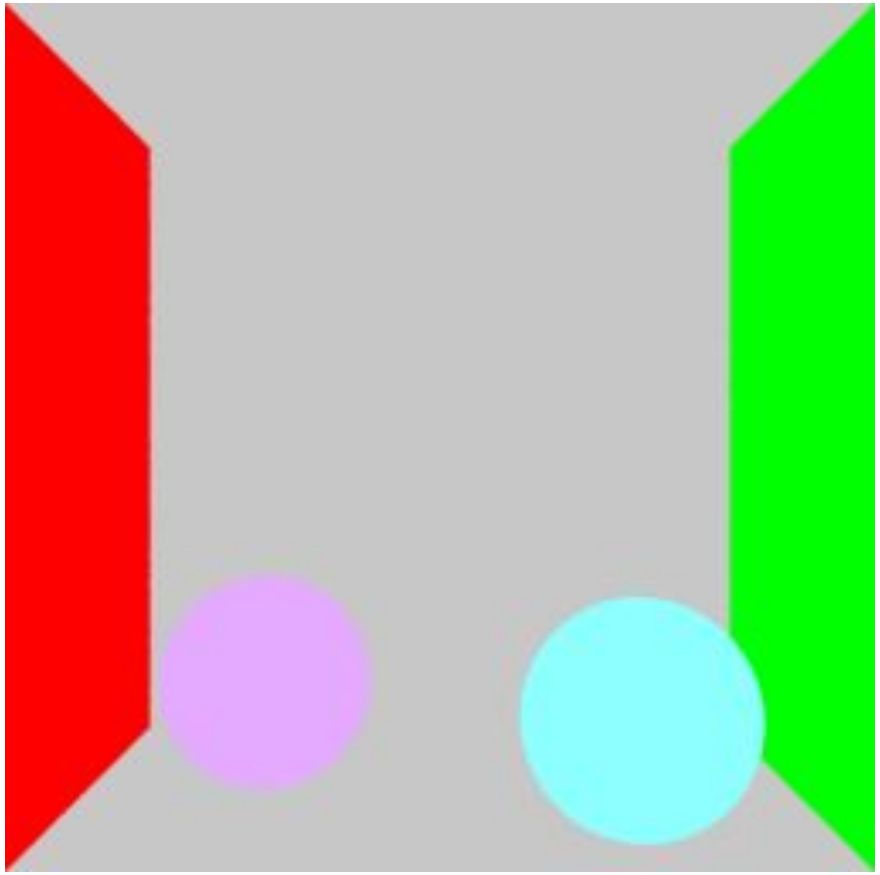
Origin	O = (0, 0, -3.5)
Left	L = (-1, 0, 0)
Up	U = (0, 1, 0)
Forward	F = (0, 0, 3)
Size	256x256 pixels

Example scene: Cornell Box

- **Results**



1 ray per pixel



64 rays per pixel

Questions

DO ASK questions, either now or after the lab

But be reasonable, please :)

pluesia@unizar.es | dsubias@unizar.es | o.pueyo@unizar.es

What to expect from this session

In the programming language of your choice, implement:

- **Scene (collection of geometry primitives)**
  - Spheres, planes, etc. (position, emission) should be hardcoded
- **Pinhole camera**
  - Generate a ray for a specific pixel
  - Intersect this ray with your scene
  - Generate an image
- Recommended deadline: **October 18th**.
  - **Extensions** (do not count towards deadline):
    - Other primitives: cones, cylinders, ellipsoids, disks or triangles
    - Acceleration structures: bounding volumes, multi-threading, etc.
    - Constructive solid geometry: google it or ask us :)