3D Graphics

Introduction

What is computer graphics

Modeling -> how do we represent stuff

Rendering -> how do we print stuff on the screen

Animation -> how do we make stuff move

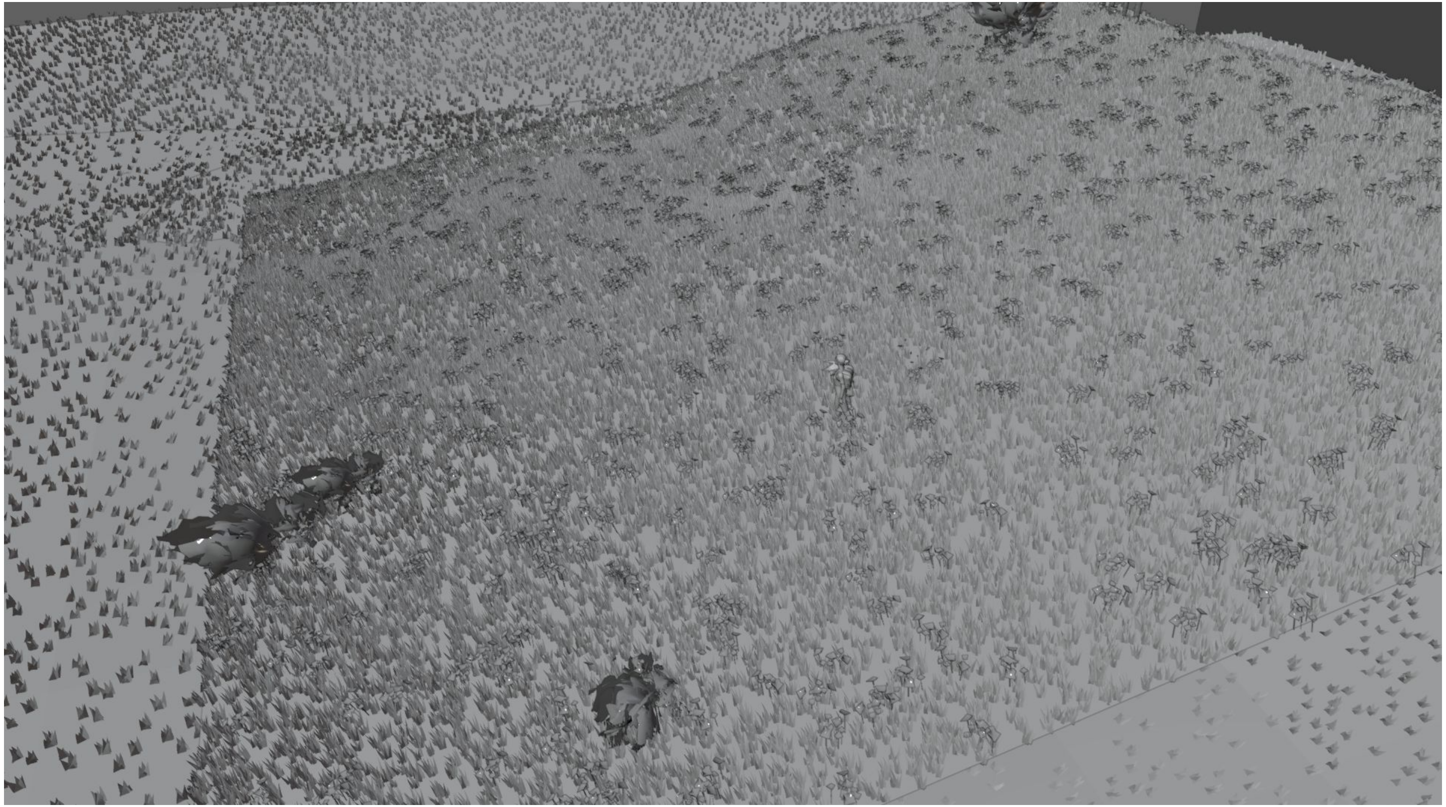
What is computer graphics

Modeling -> how do we represent stuff

Rendering -> how do we print stuff on the screen

Animation -> how do we make stuff move

Rendering : Transforming a **scene** into an **image**



Rendering : Transforming a **scene** into an **image**

Red Autumn Forest

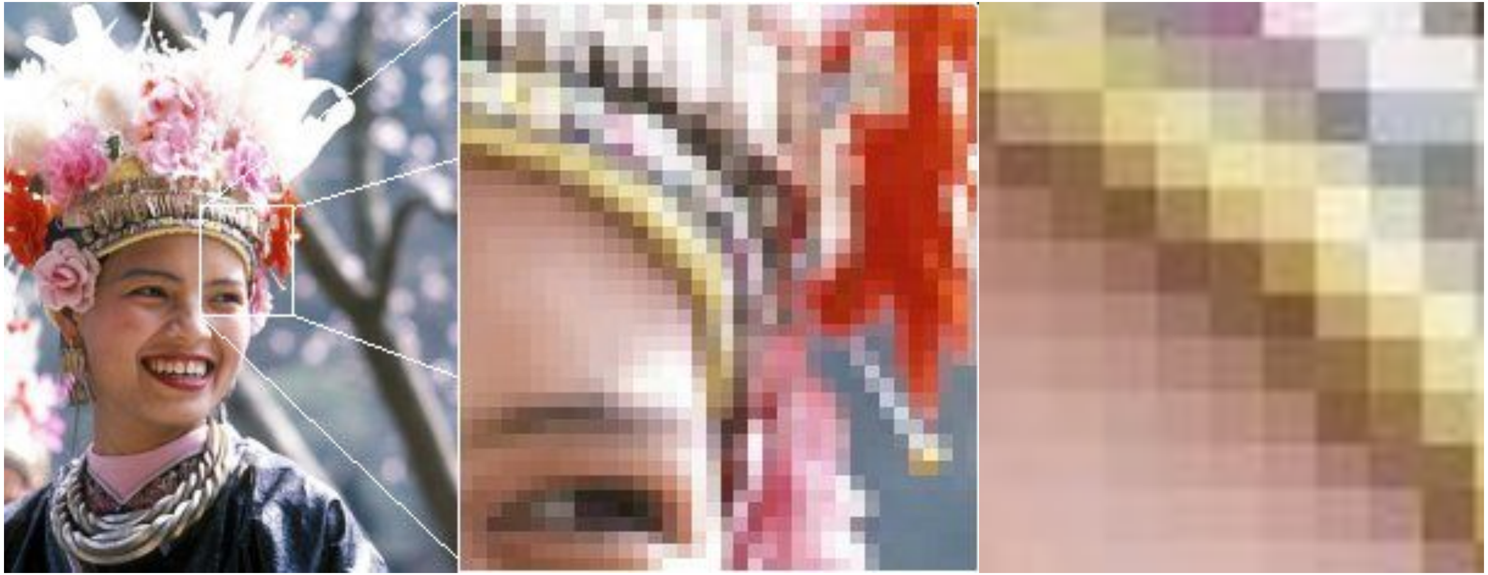
By Robin Tran

What is an image

Images

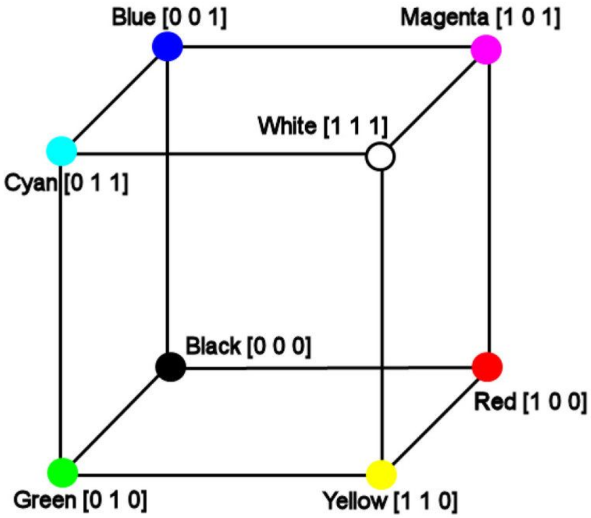
2D array of pixels

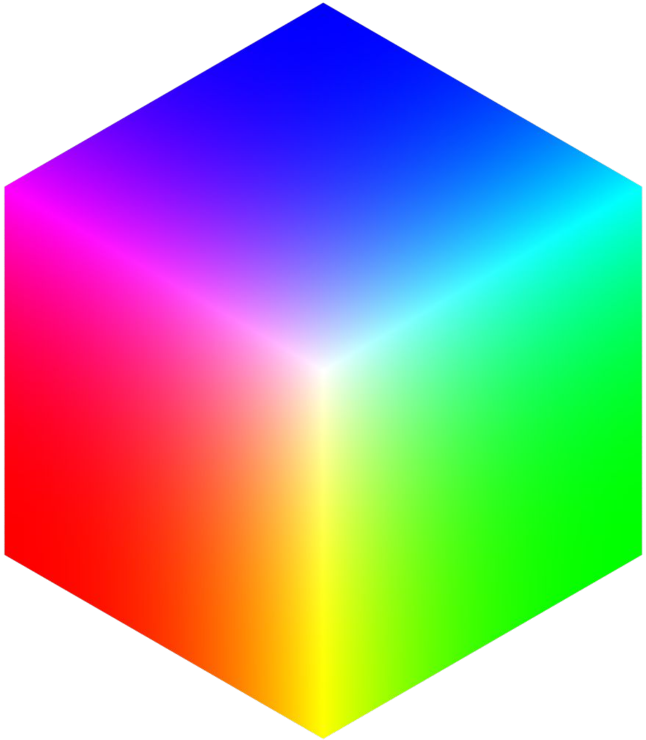
Each Pixel stores a color



Color representation : Red Green and Blue

Representing color on three axis :





Question

Why did we choose red green and blue ?

(1 minute alone)

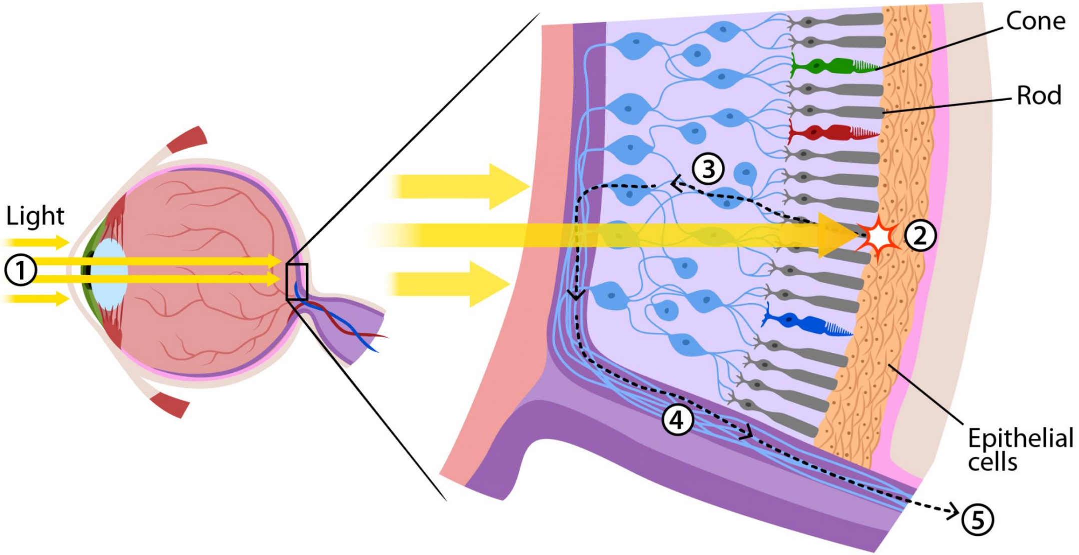
(2 minutes with your neighbors)

(5 minutes with the whole group)

Human vision

Cone cell - > detect color

Rod cell - > detect intensity

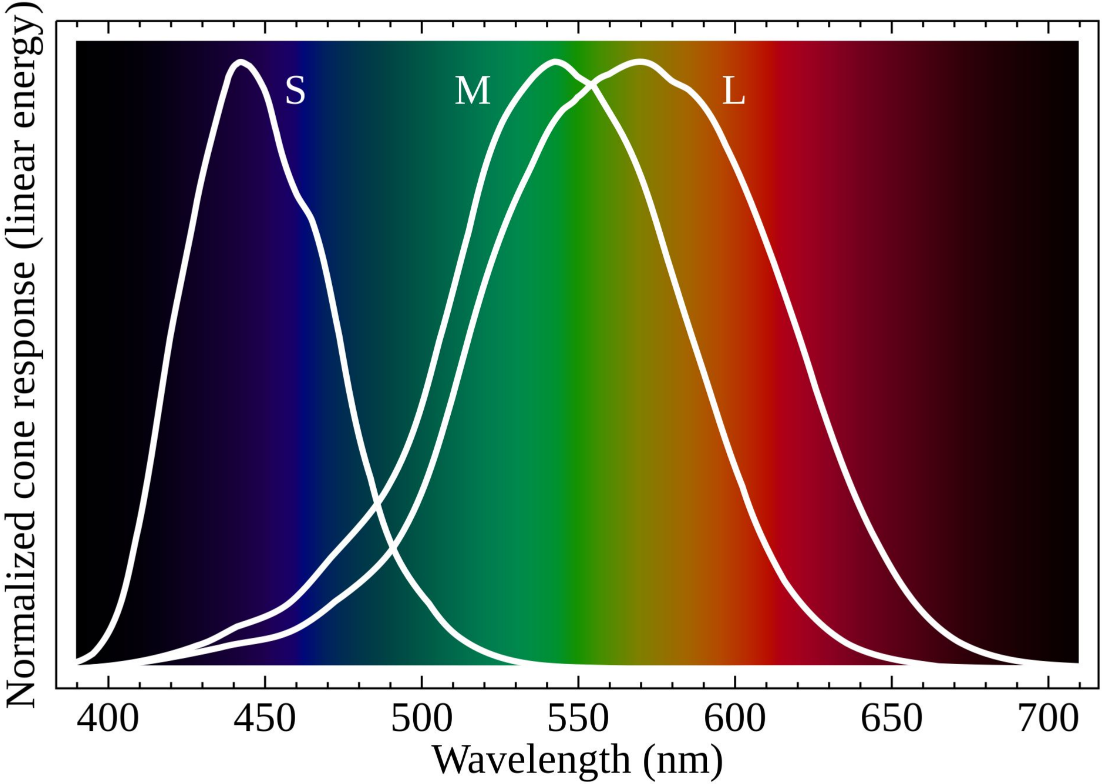


Human vision

A type of cone for blue

A type of cone for green

A type of cone for red

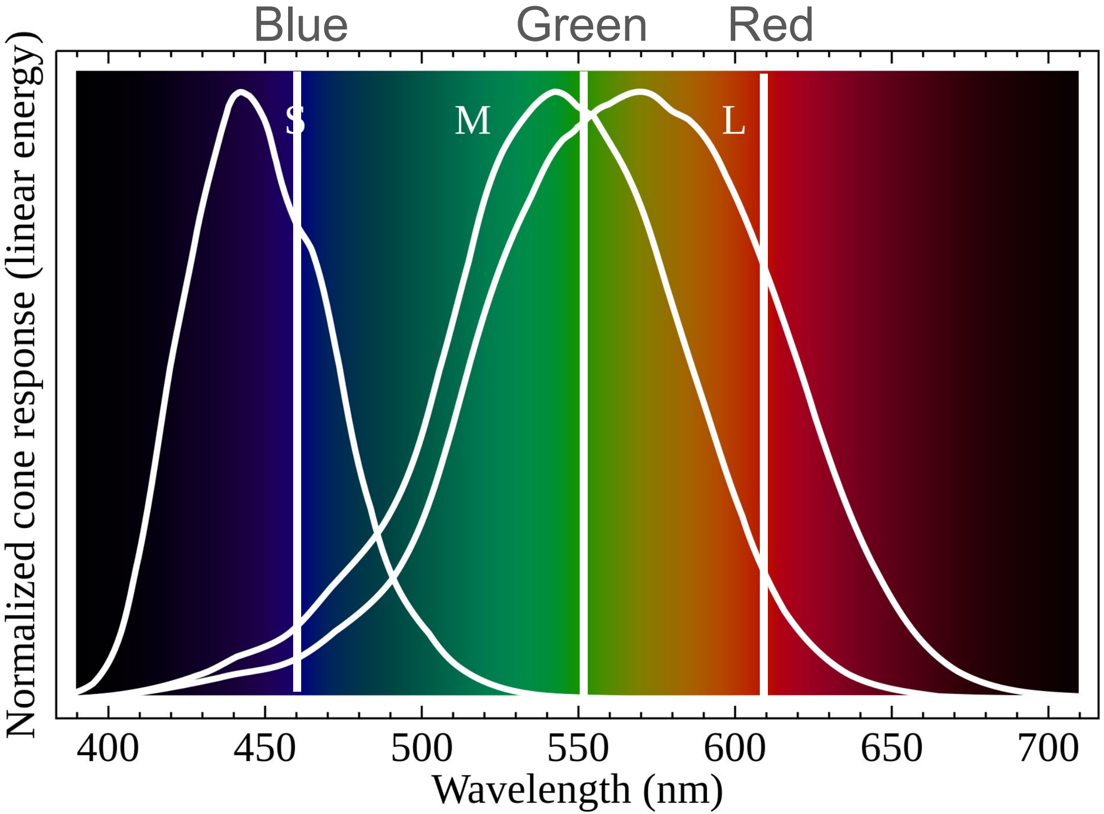


Human vision

A type of cone for blue

A type of cone for green

A type of cone for red

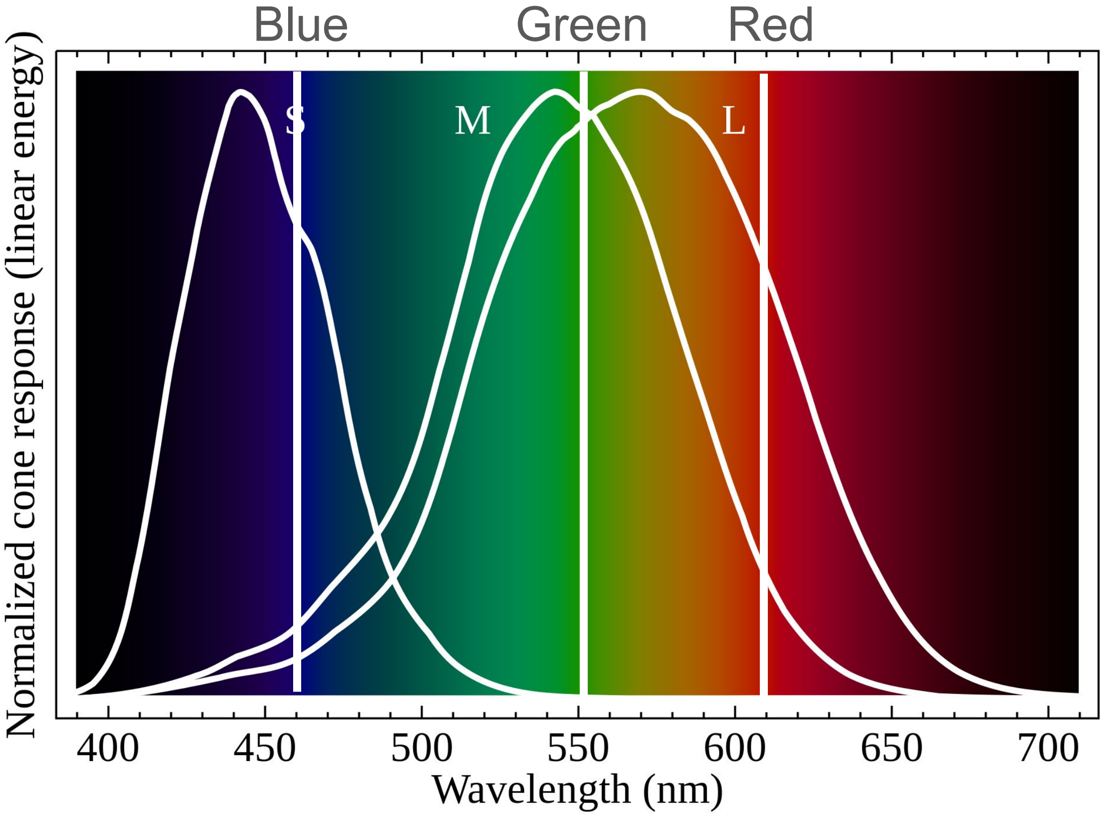


Human vision

~~A type of cone for blue~~

~~A type of cone for green~~

~~A type of cone for red~~



Human vision

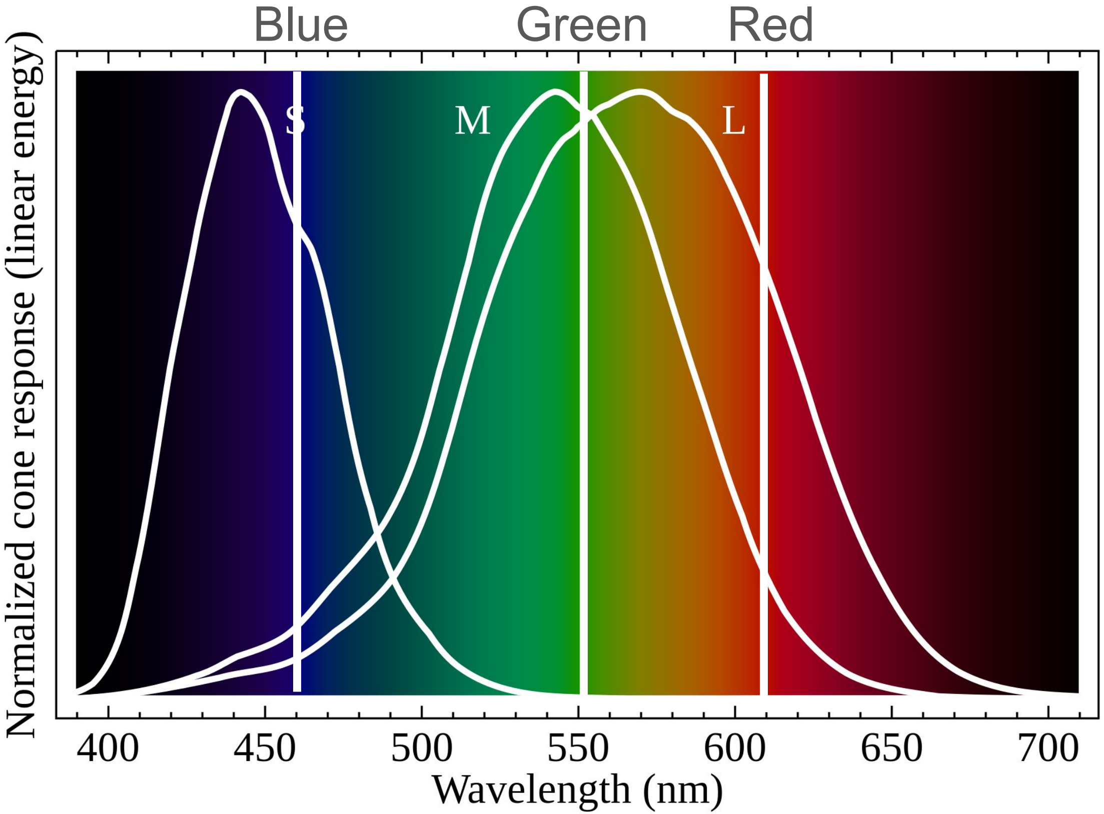
~~A type of cone for blue~~

~~A type of cone for green~~

~~A type of cone for red~~

Color Photography -> 1926

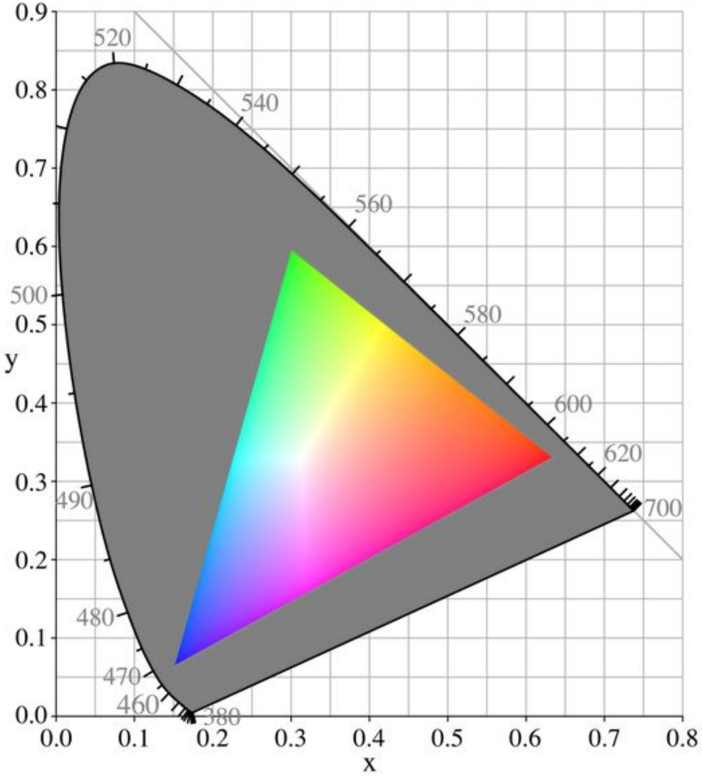
Cone cell Discovery - > 1956



RGB can represent all colors ?

Gamut: the subset of color achievable by a representation

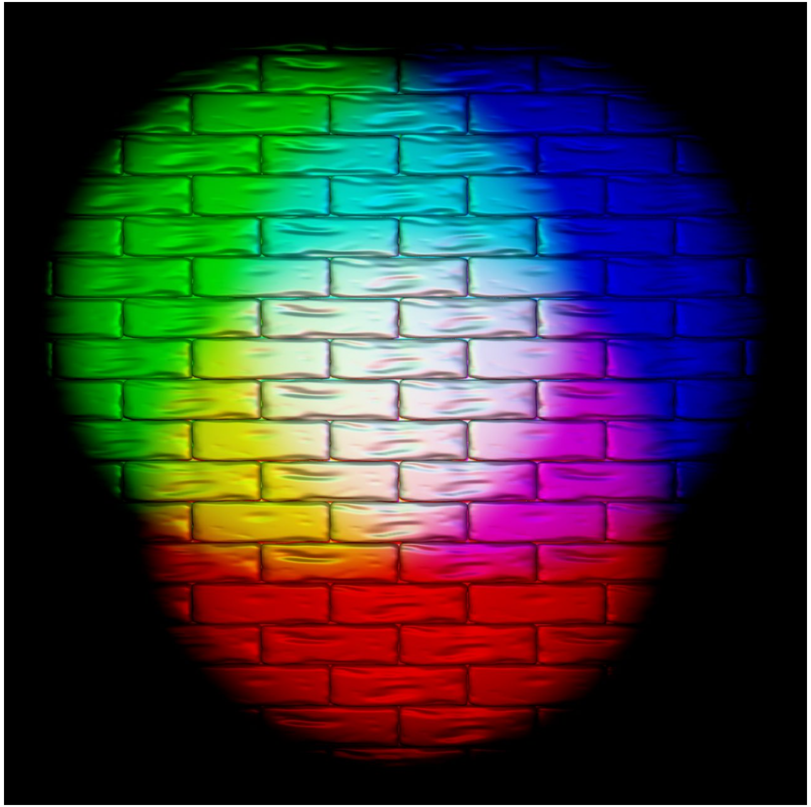
RGB only represent a subset of the visible color



Visible color vs RGB color gamut

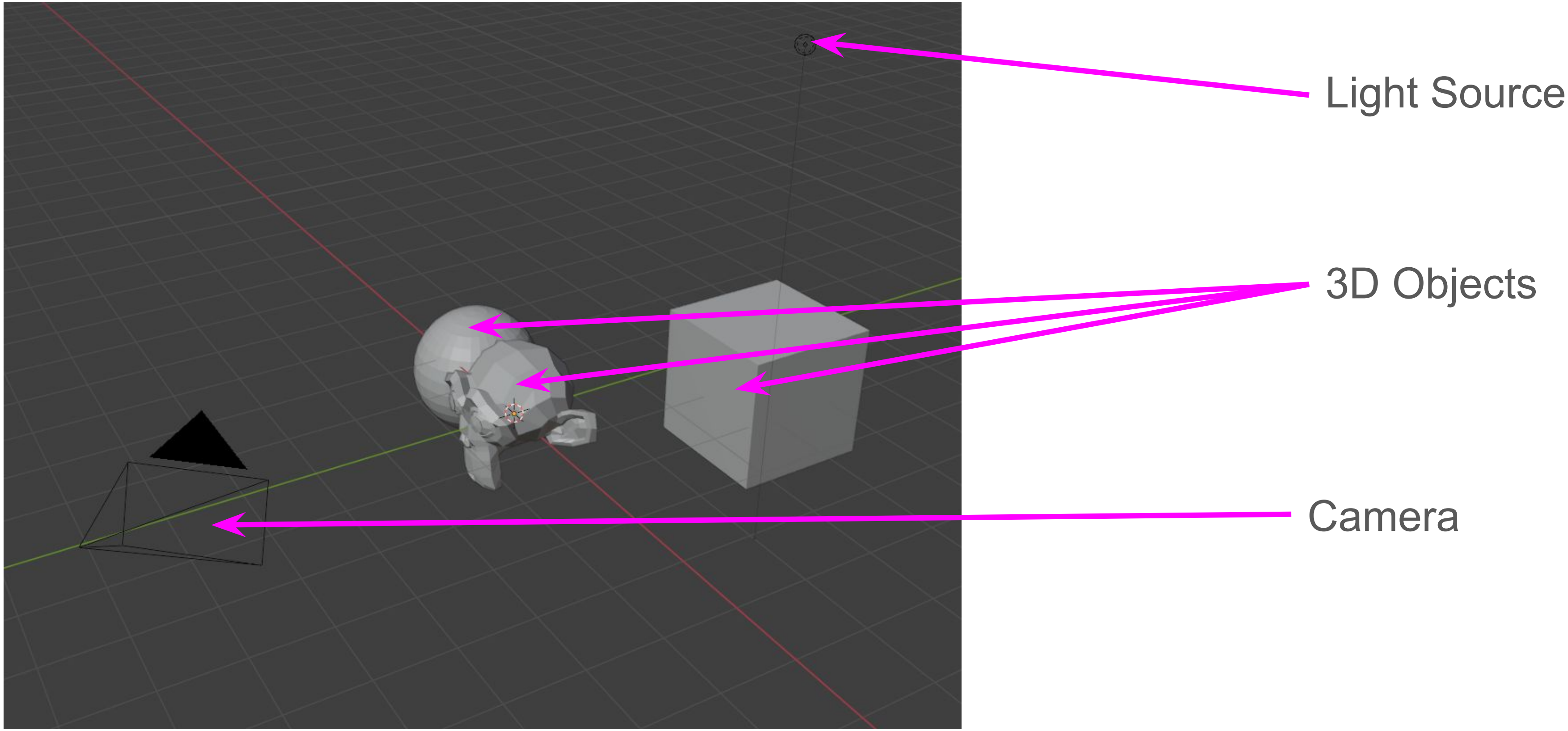
Additive Color

We emit light from the screen
So color are added



What is a scene ?

Scene

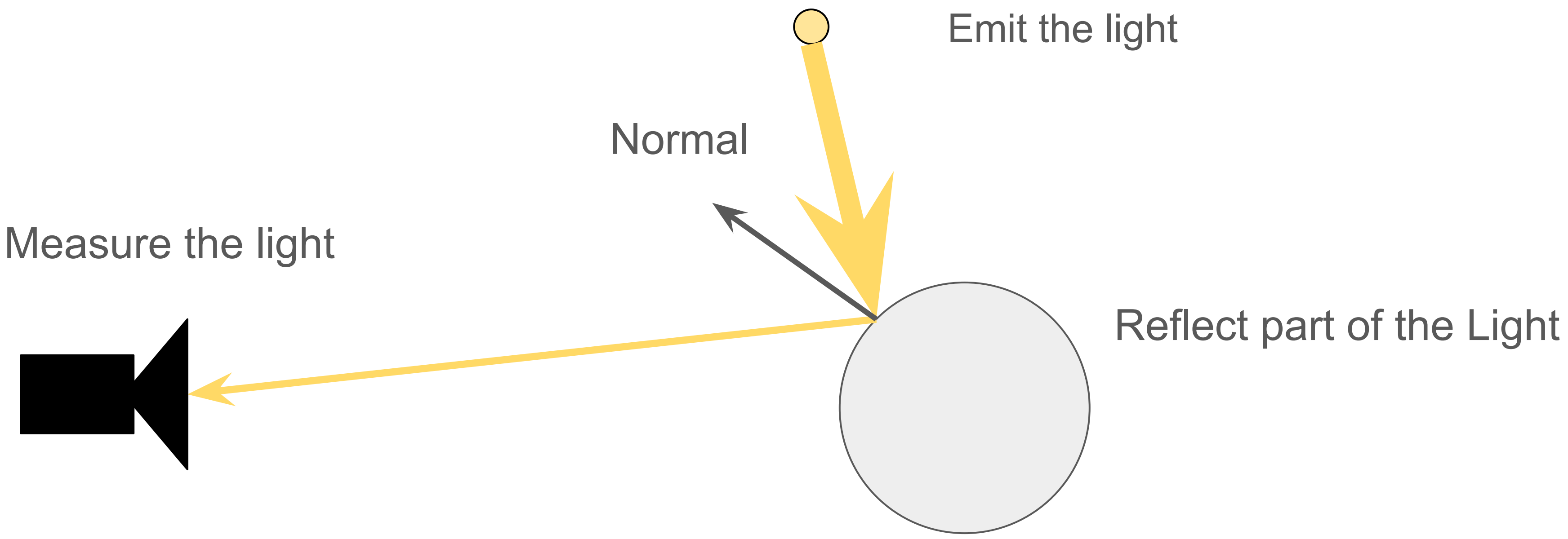


Light Source

3D Objects

Camera

Scene


Emit the light

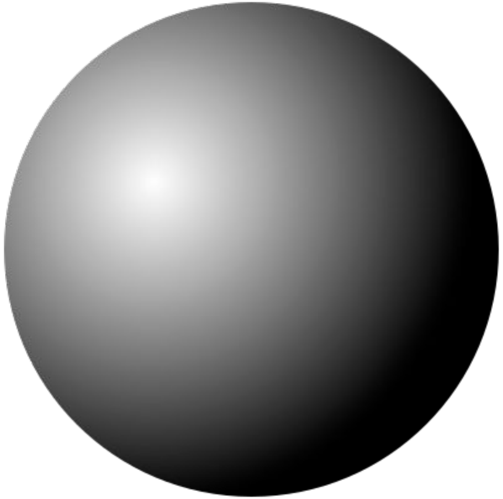
Normal

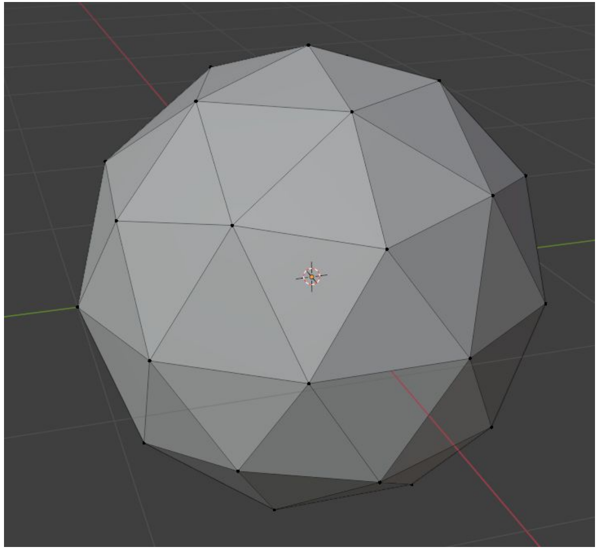
Measure the light

Reflect part of the Light

Scene : 3D Mesh

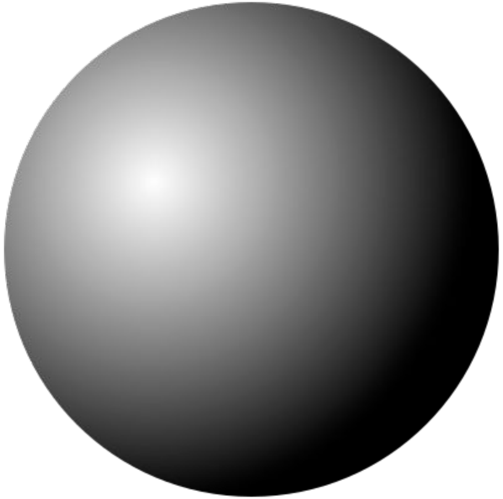
Key idea : approximating surfaces using triangles

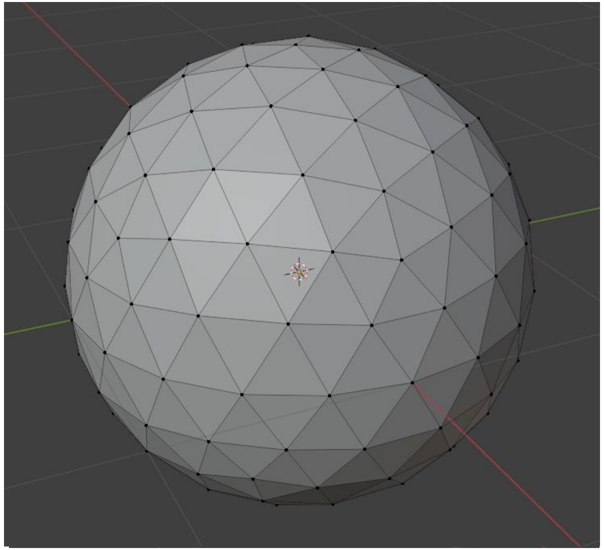




Scene : 3D Mesh

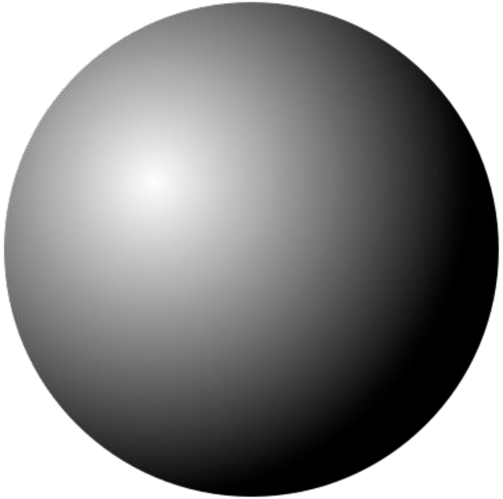
Key idea : approximating surfaces using triangles

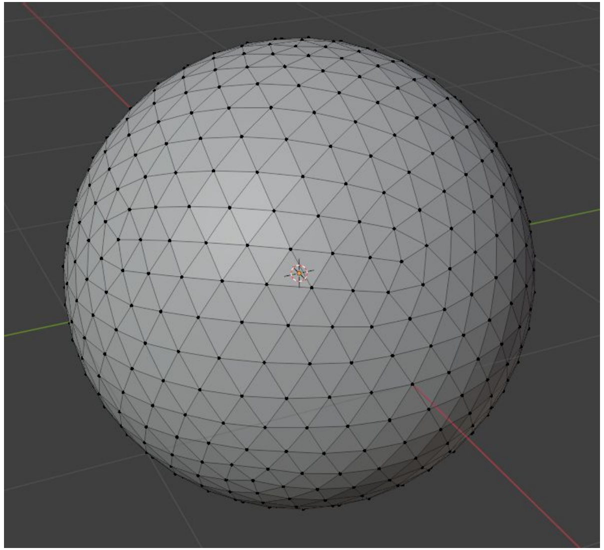




Scene : 3D Mesh

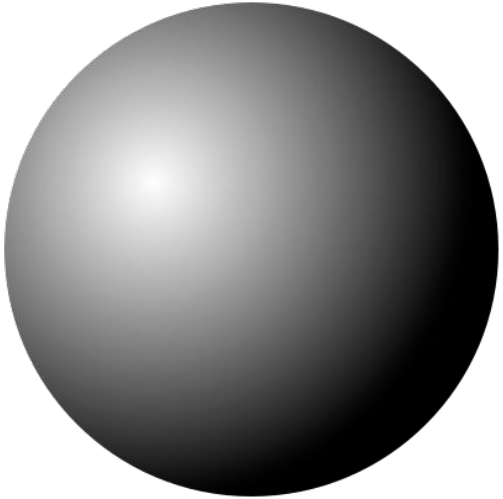
Key idea : approximating surfaces using triangles

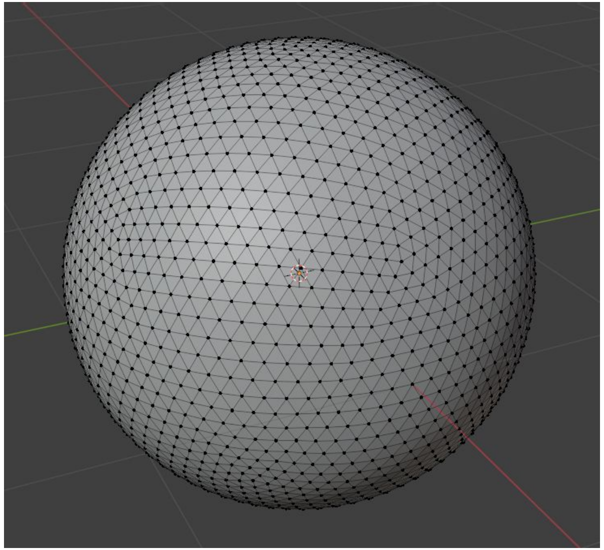




Scene : 3D Mesh

Key idea : approximating surfaces using triangles





Question

Why do we use triangles (and not quads, circles or other primitives) ?

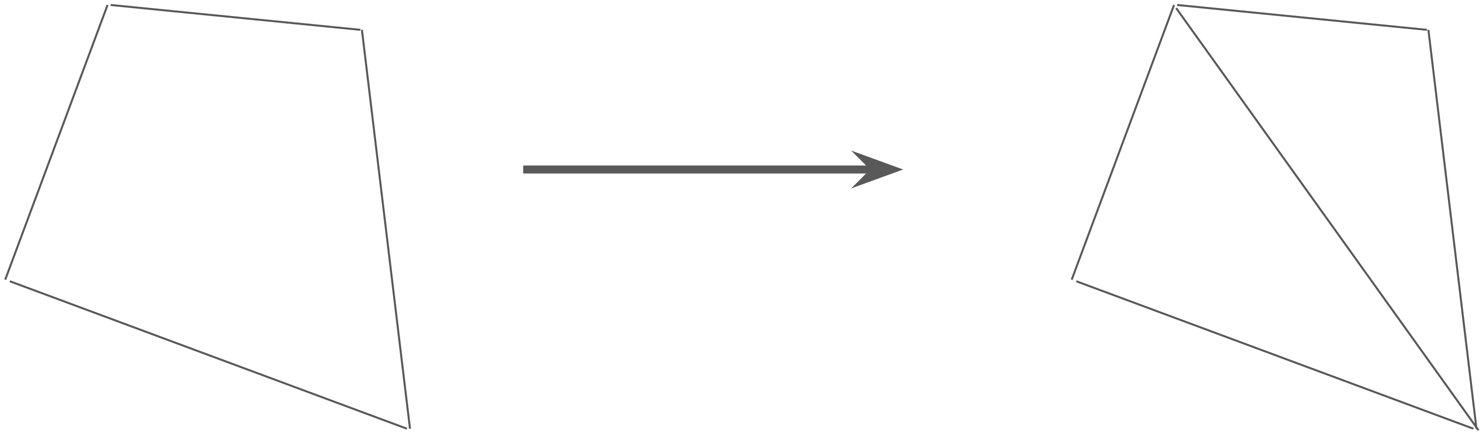
(1 minute alone)

(2 minutes with your neighbors)

(5 minutes with the whole group)

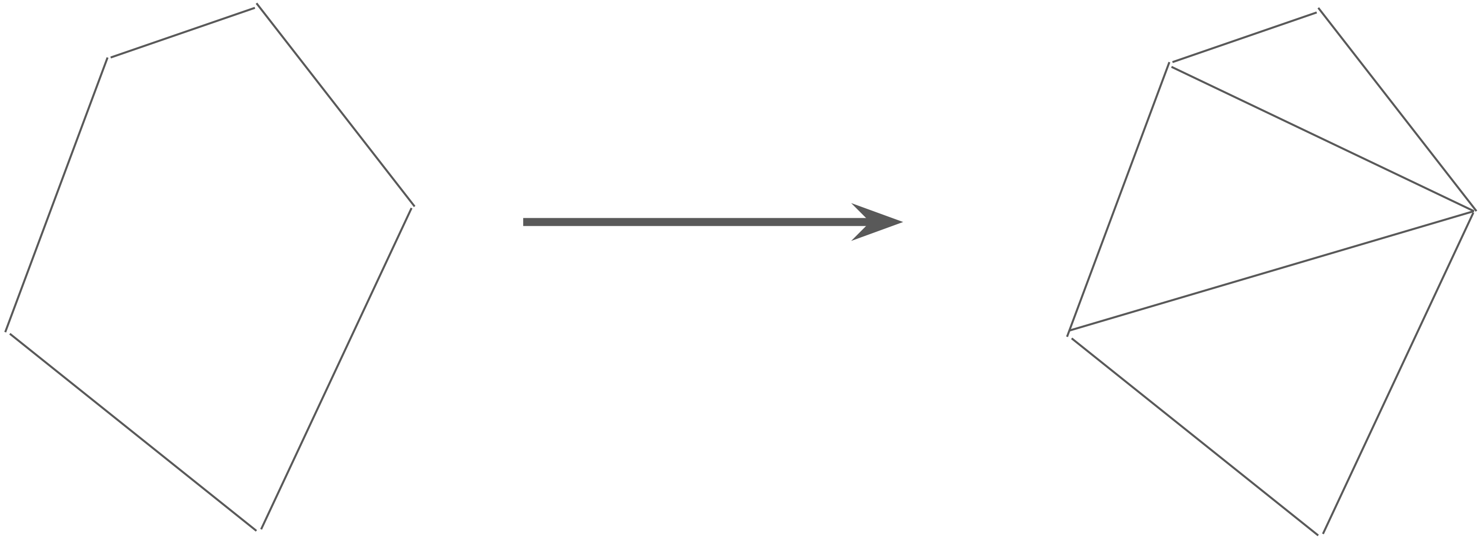
Everything is triangles

A quad is two triangle

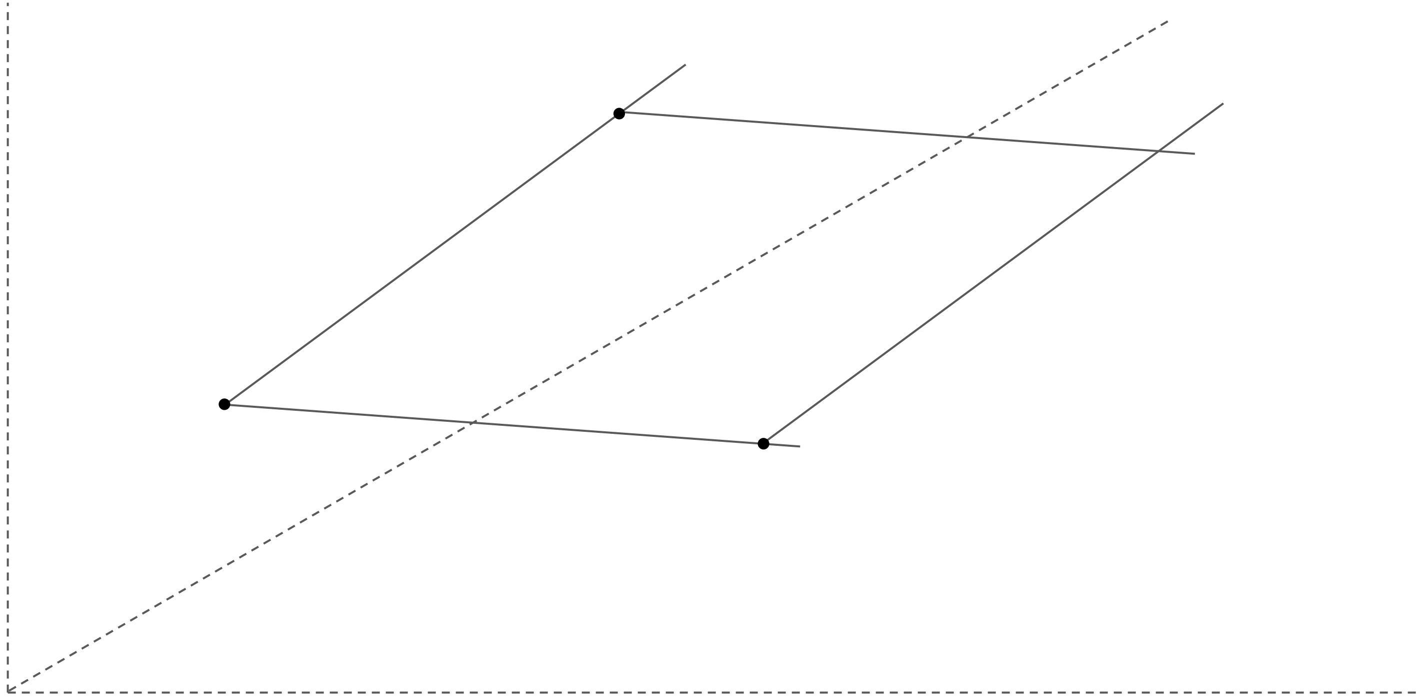


Everything is triangles

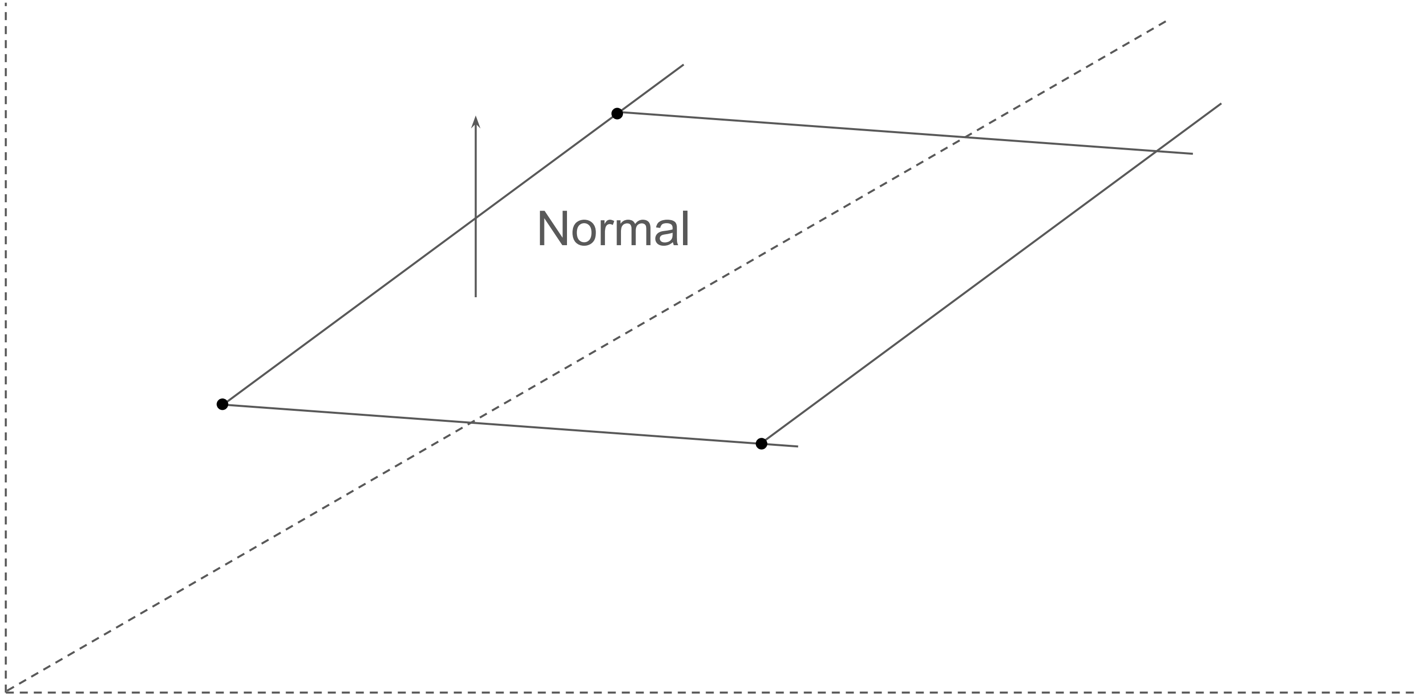
A quad is two triangle



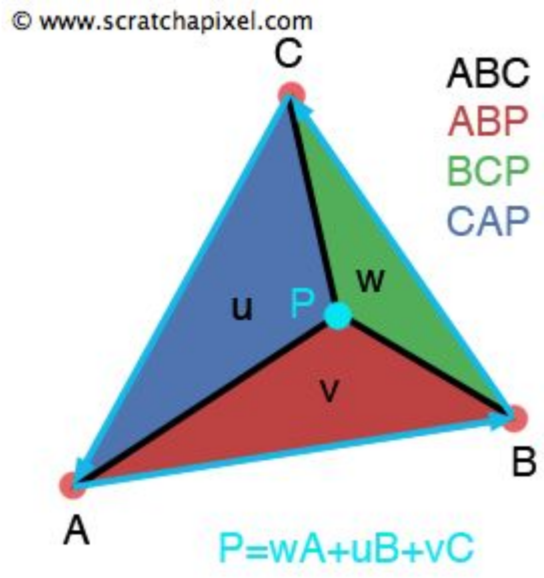
Triangle : Three point make a plane



Triangle : Three point make a plane



Triangle : Barycentric coordinates



Each point in the triangle is a linear composition of the three vertices

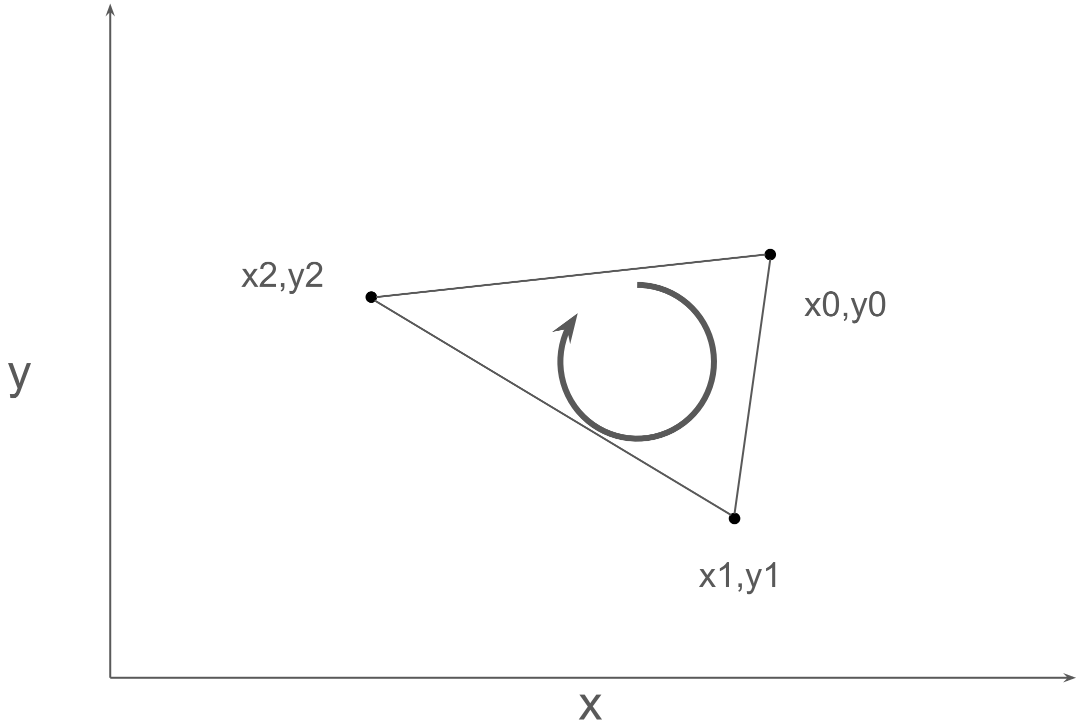
Triangles and how to store them

Mesh representation : Triangle soup

Each triangle is store as a set of three coordinates in the counter clockwise order

Example in 2D : one triangle

{x0,y0,
x1,y1,
x2,y2}

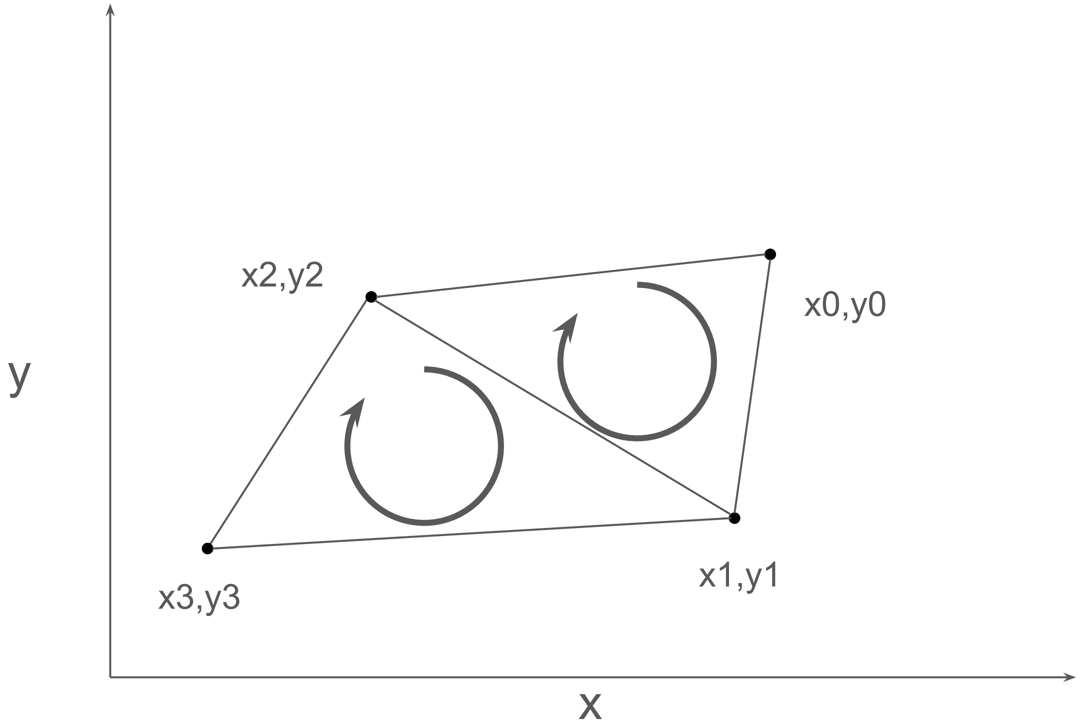


Mesh representation : Triangle soup

Each triangle is store as a set of three coordinates in the counter clockwise order

Example in 2D : one triangle

$$\left.\begin{array}{l}\{x0,y0,\\ x1,y1,\\ x2,y2,\end{array}\right\}\text{Triangle 1}$$
$$\left.\begin{array}{l}x2,y2,\\ x1,y1,\\ x3,y3\}\end{array}\right\}\text{Triangle 2}$$



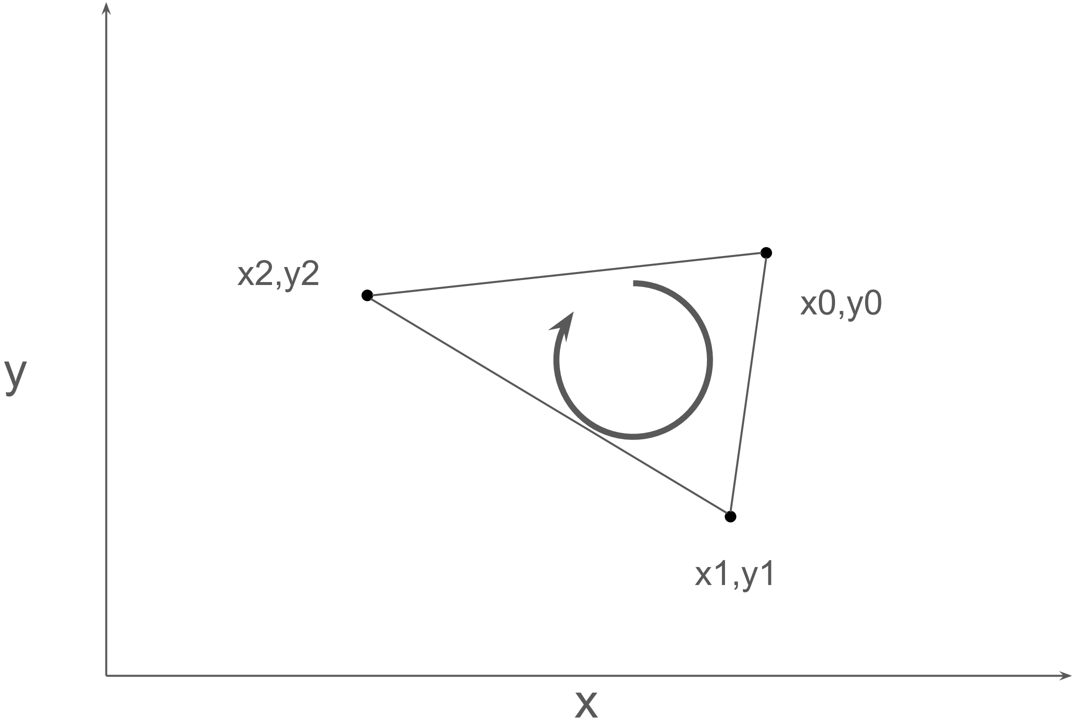
Mesh representation : Indexed Triangle

Each triangle is store as a set of three coordinates in the counter clockwise order

Example in 2D : one triangle

Vertices list
{x0,y0,
x1,y1,
x2,y2}

Indices list
{0,1,2}



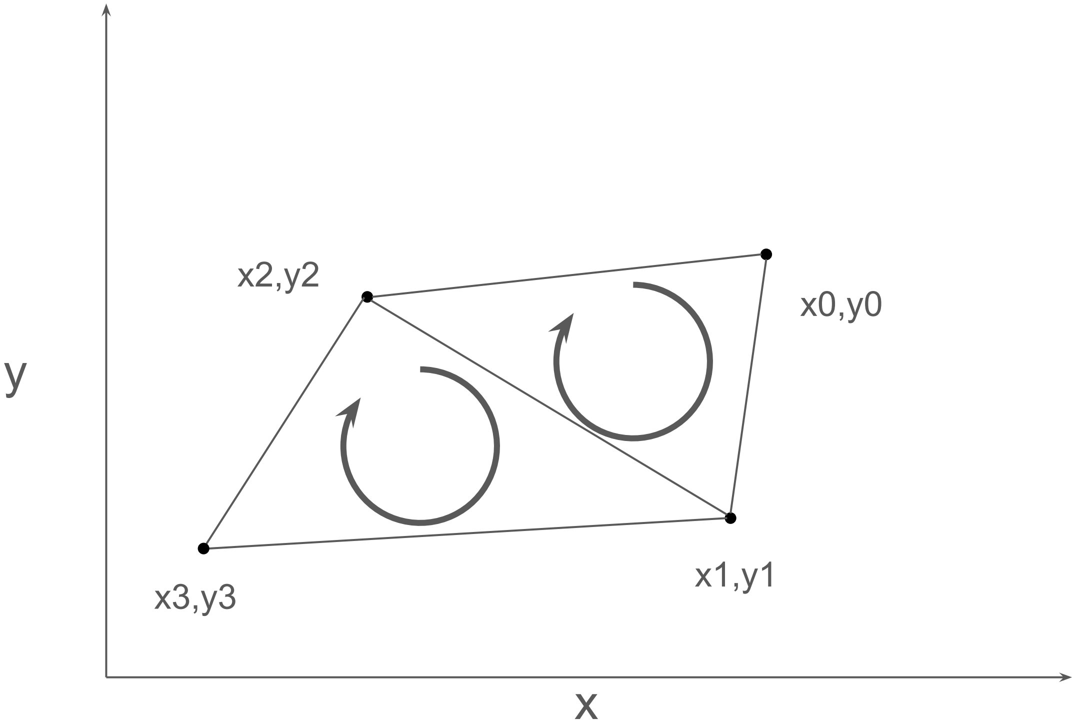
Mesh representation : Indexed Triangle

Each triangle is store as a set of three coordinates in the counter clockwise order

Example in 2D : one triangle

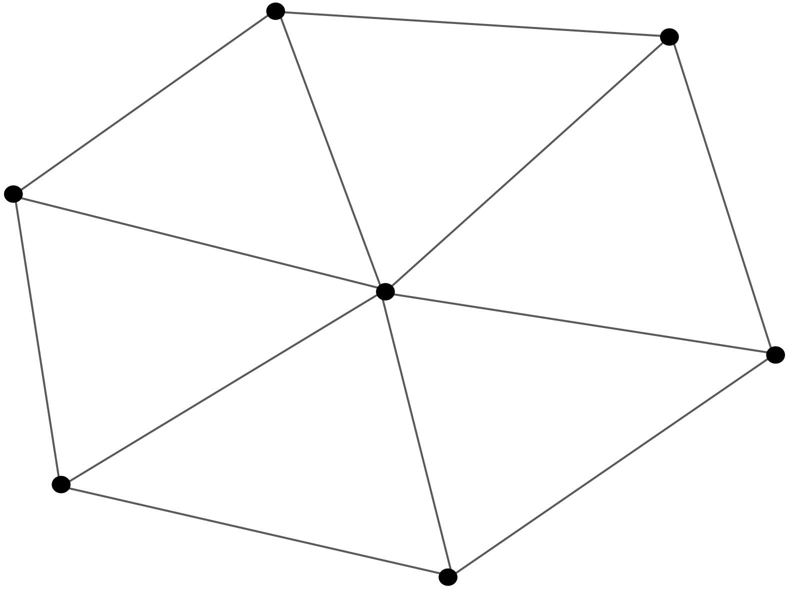
Vertices list
{x0,y0,
x1,y1,
x2,y2,
x3,y3}

Indices list
{
0,1,2, -> triangle 1
2,1,3   -> triangle 2
}



Question

In the following triangles estimate the memory consumption of storing them as a soup and as an indexed list :



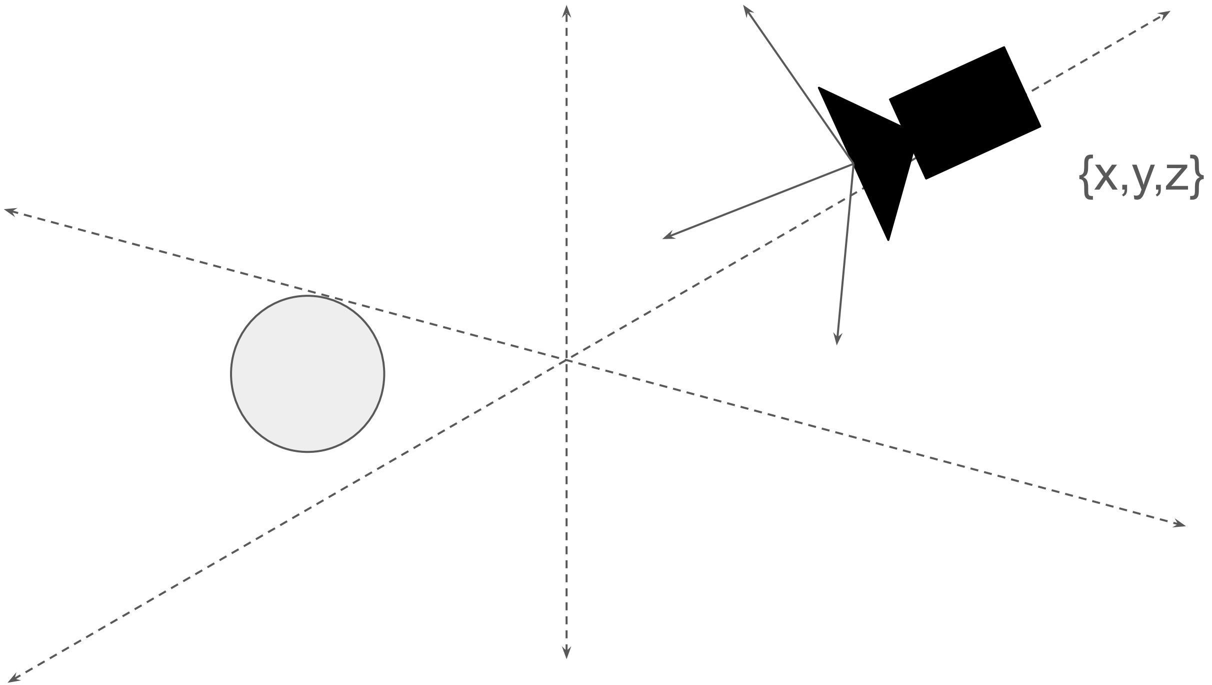
Infos :

6 triangles,
7 vertices

Camera

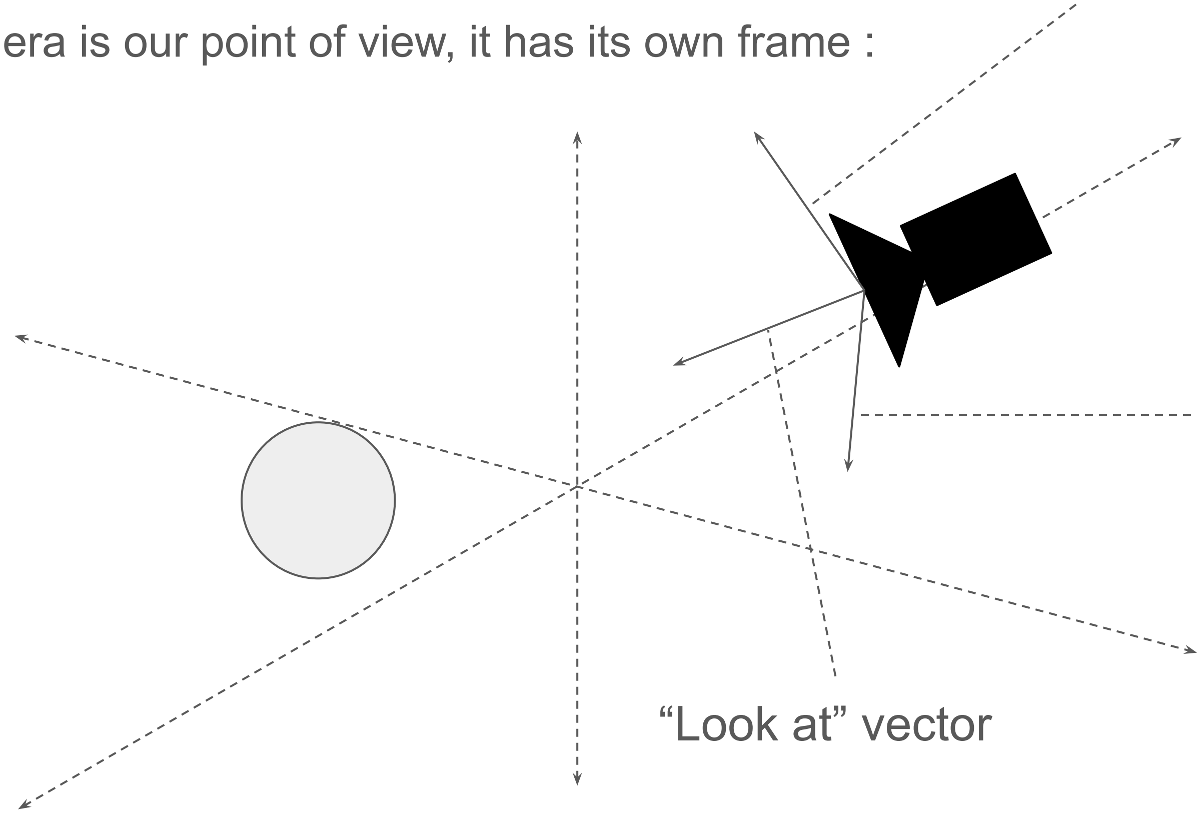
Scene : Camera

The camera is our point of view, it has a position in the scene:



Scene : Camera

The camera is our point of view, it has its own frame :



“Up” vector

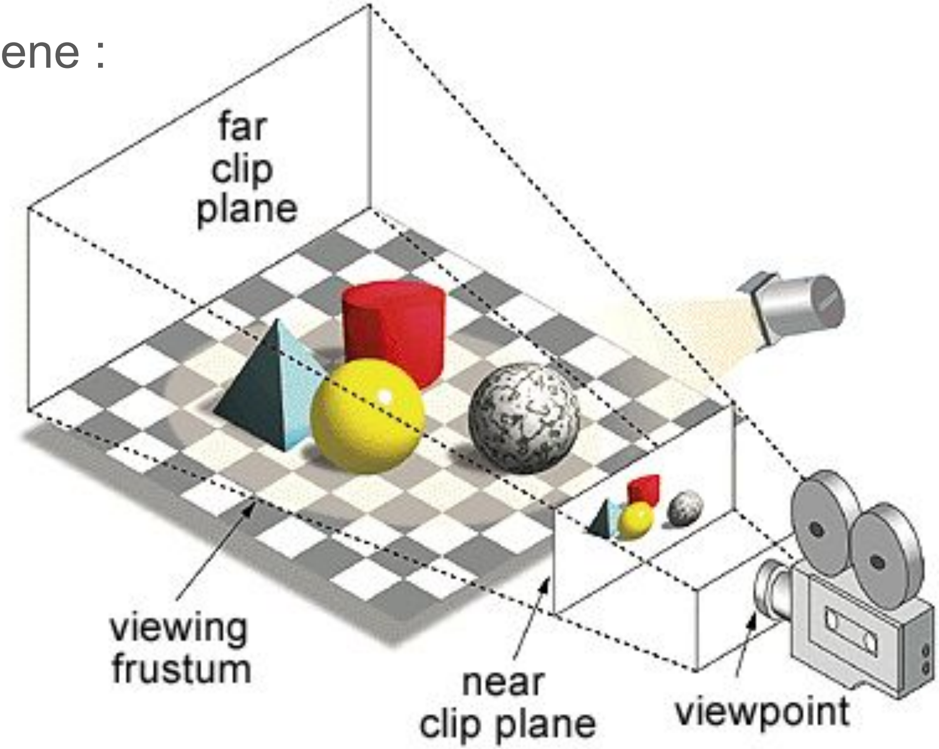
“Right” vector

“Look at” vector

Scene: Camera

Frustum : the visible part of the scene :

- Near plane
- Far plane
- Aspect ratio
- Field of View


far
clip
plane
viewing
frustum
near
clip plane
viewpoint