

# **Recap Visión Artificial**

## Visión por Computador II

# Contenido

1. Transformaciones geométricas
2. Formación de la imagen
3. Pinhole cámara
4. Lentes
5. Operadores de punto

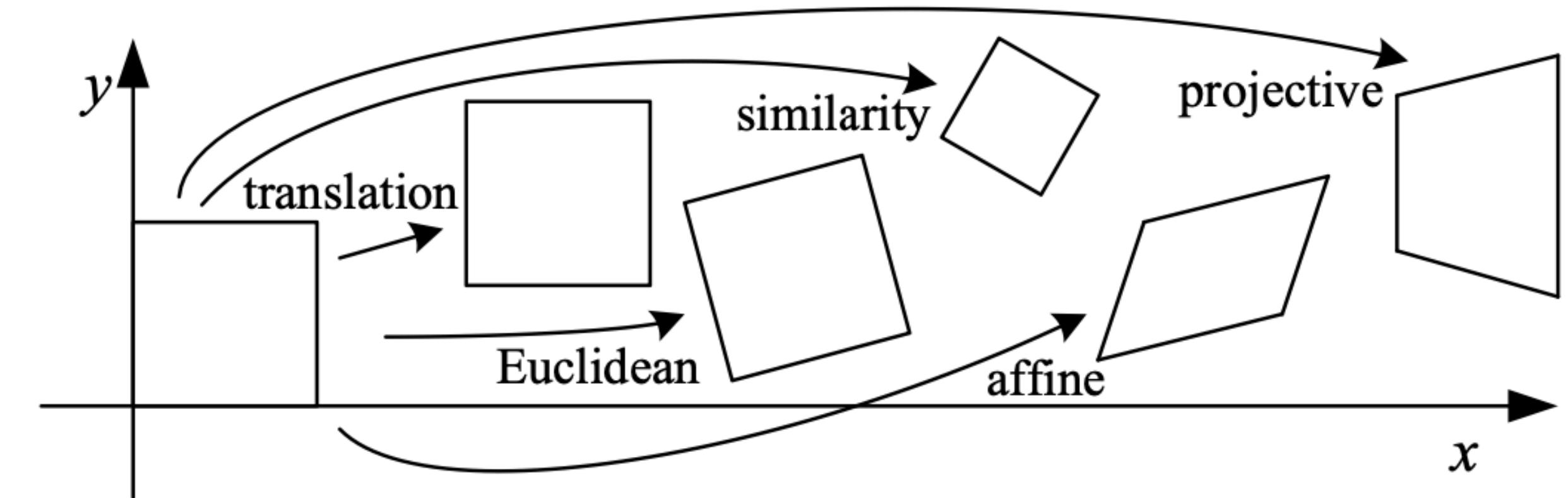


¿Son los autos del mismo tamaño en la imagen?

# Transformaciones geométricas

## Transformaciones 2D

- Posición de los objetos  
posición sus puntos
- Aplicar la transformación a  
todos sus puntos



Conjunto básico de transformaciones 2D. Tomado de Computer Vision: Algorithms and Applications

# Translación

Trasladar el punto  $P$  a  $P'$ :

$$P(x, y) \rightarrow P'(x', y')$$

Movimiento paralelo a los ejes

$$x' = x + d_x$$

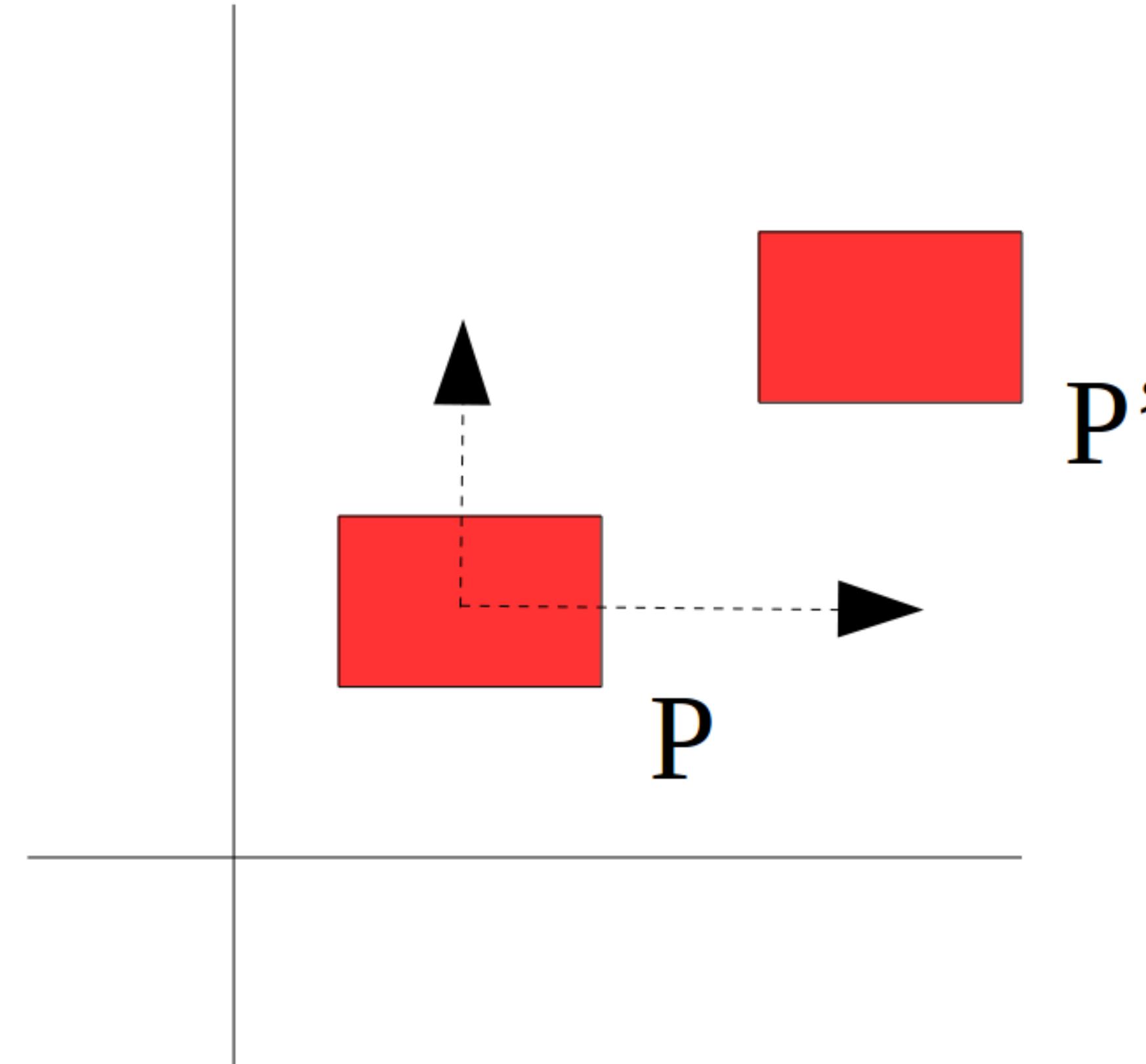
$$y' = y + d_y$$

Definidos como vectores

$$P = \begin{bmatrix} x \\ y \end{bmatrix}, P' = \begin{bmatrix} x' \\ y' \end{bmatrix}, T = \begin{bmatrix} d_x \\ d_y \end{bmatrix}$$

Entonces:

$$P' = P + T$$



# Escala desde el origen

Definido el punto  $P(x, y)$

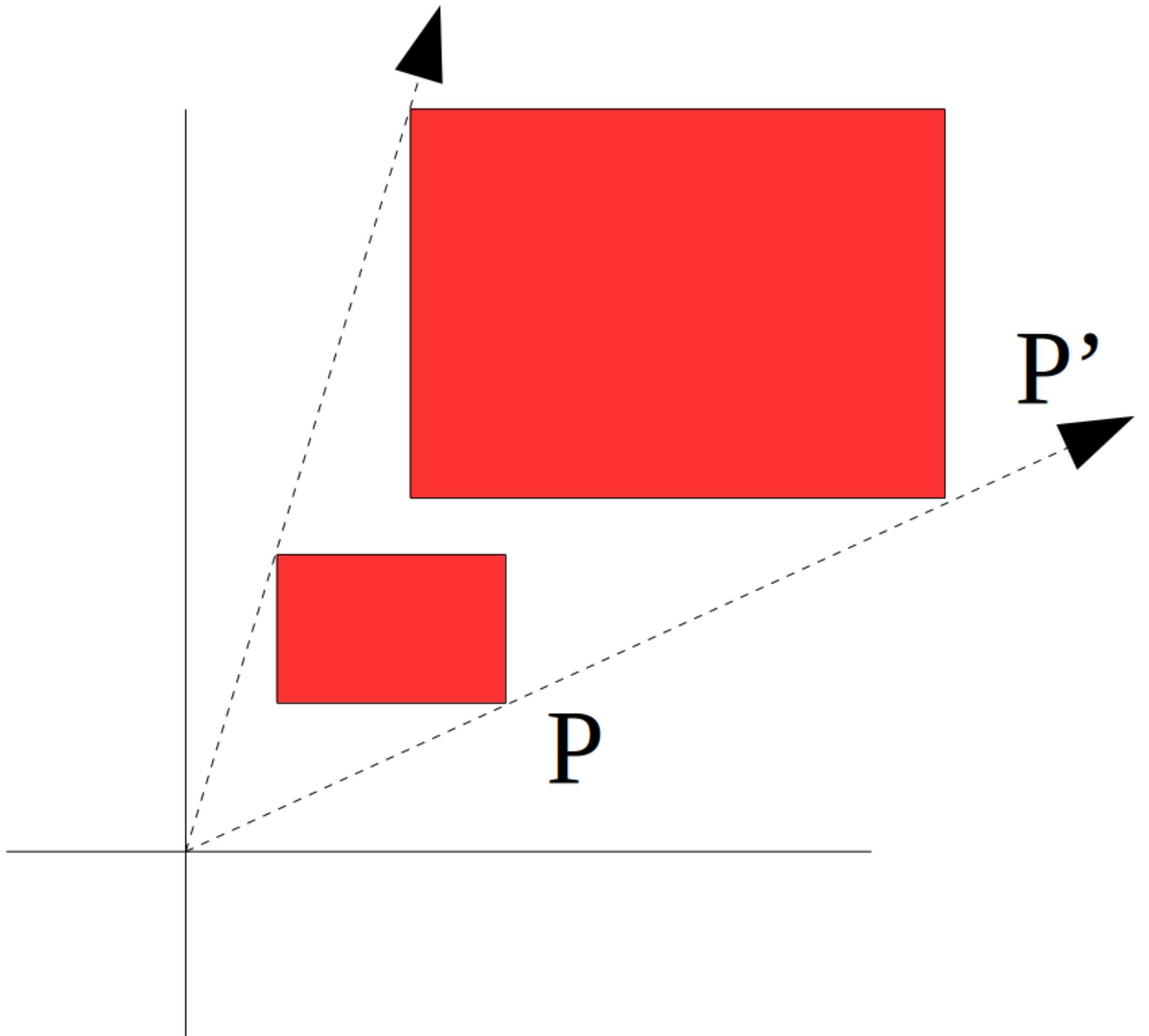
Escalarlo por los factores  $s_x$  y  $s_y$

$$x' = s_x x$$

$$y' = s_y y$$

Como operación matricial:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$



# Rotaciones desde el origen

Un punto  $P(x, y)$  es rotado un ángulo  $\theta$

Posiciones actual de  $P$

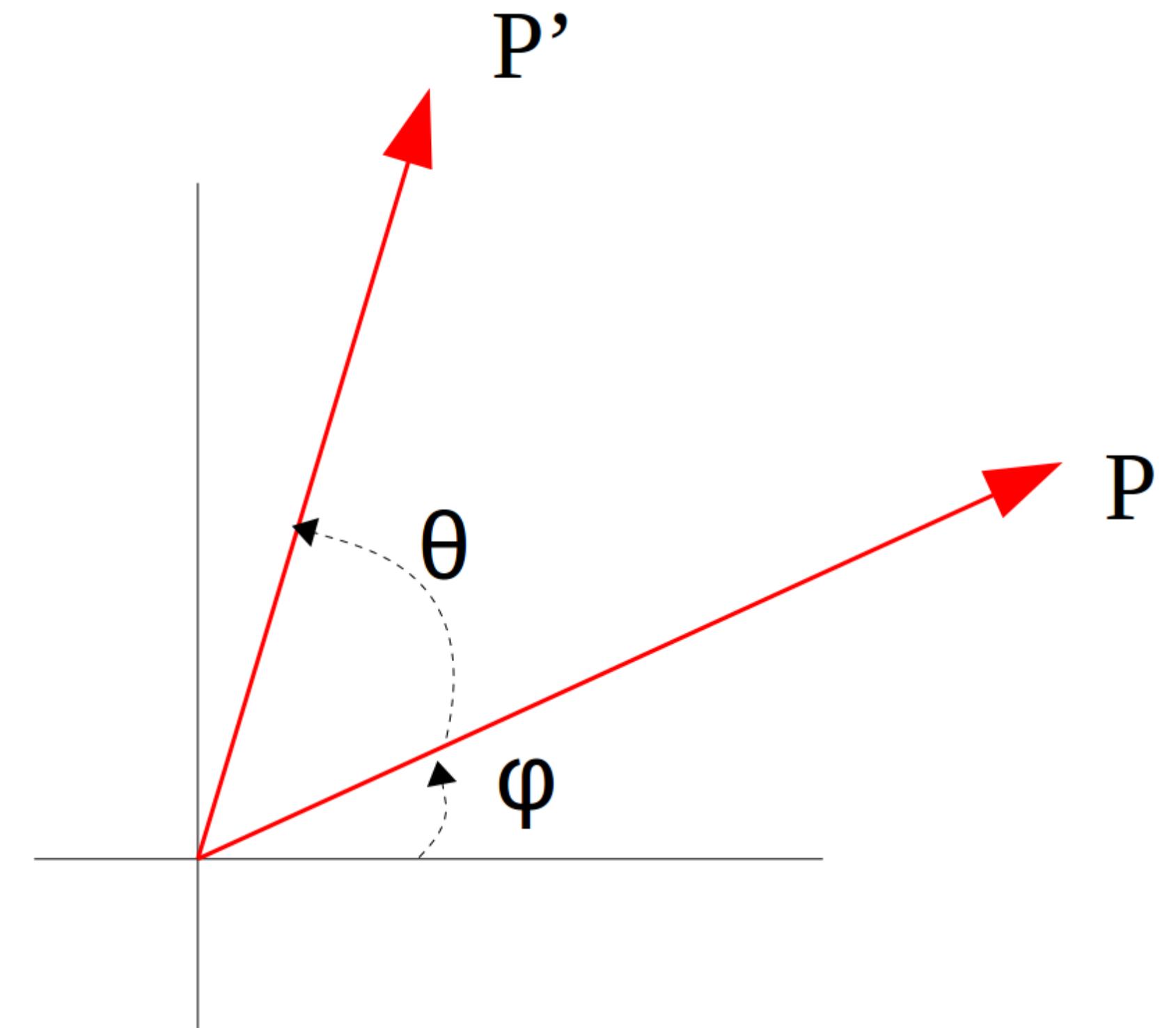
$$x = r \cos \phi$$

$$y = r \sin \phi$$

Posiciones final:

$$x' = r \cos(\theta + \phi)$$

$$y' = r \sin(\theta + \phi)$$



# Rotaciones desde el origen

Identidad de ángulos dobles

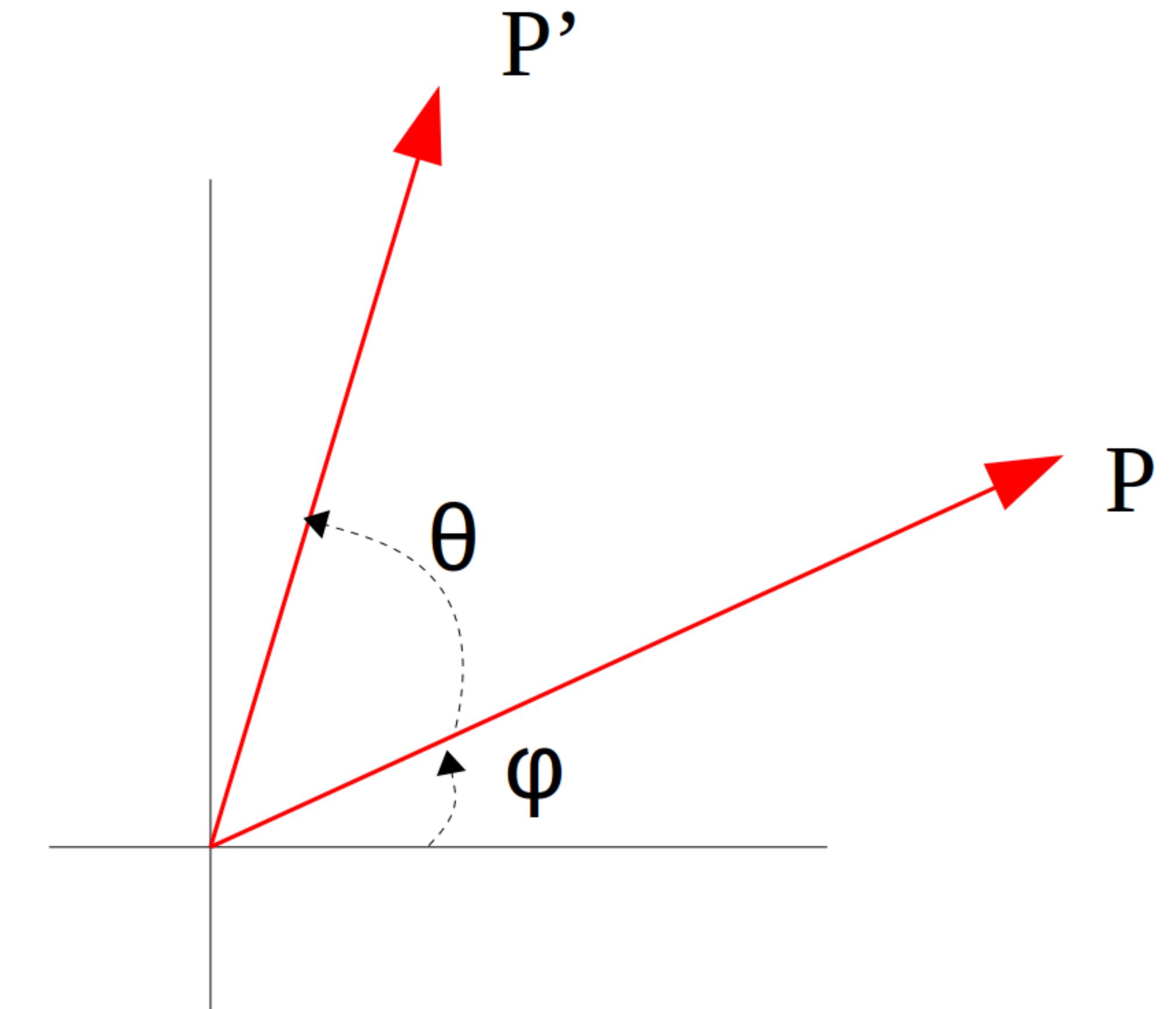
$$y' = r \cos(\phi) \sin(\theta) + r \sin(\phi) \cos(\theta)$$

$$x' = r \cos(\phi) \cos(\theta) - r \sin(\phi) \sin(\theta)$$

Sustituyendo  $x$  y  $y$

$$x' = x \cos(\theta) - y \sin(\theta)$$

$$y' = x \sin(\theta) + y \cos(\theta)$$



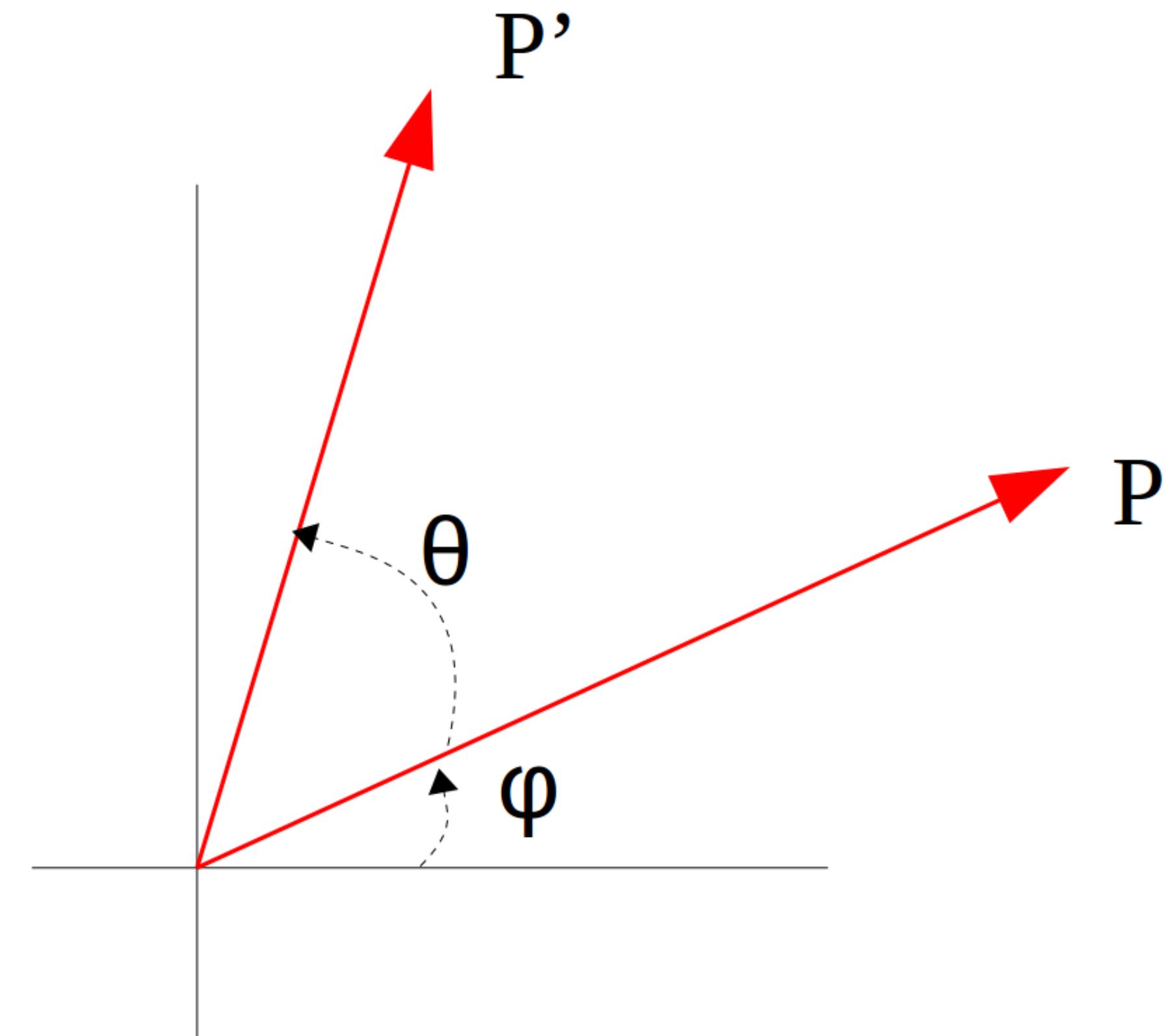
**identidades trigonométricas de suma de ángulos:**

<https://es.khanacademy.org/math/precalculus/x9e81a4f98389efdf:trig/x9e81a4f98389efdf:angle-addition/v/trigonometry-identity-review-fun>

# Rotaciones desde el origen

Forma de operación matricial

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

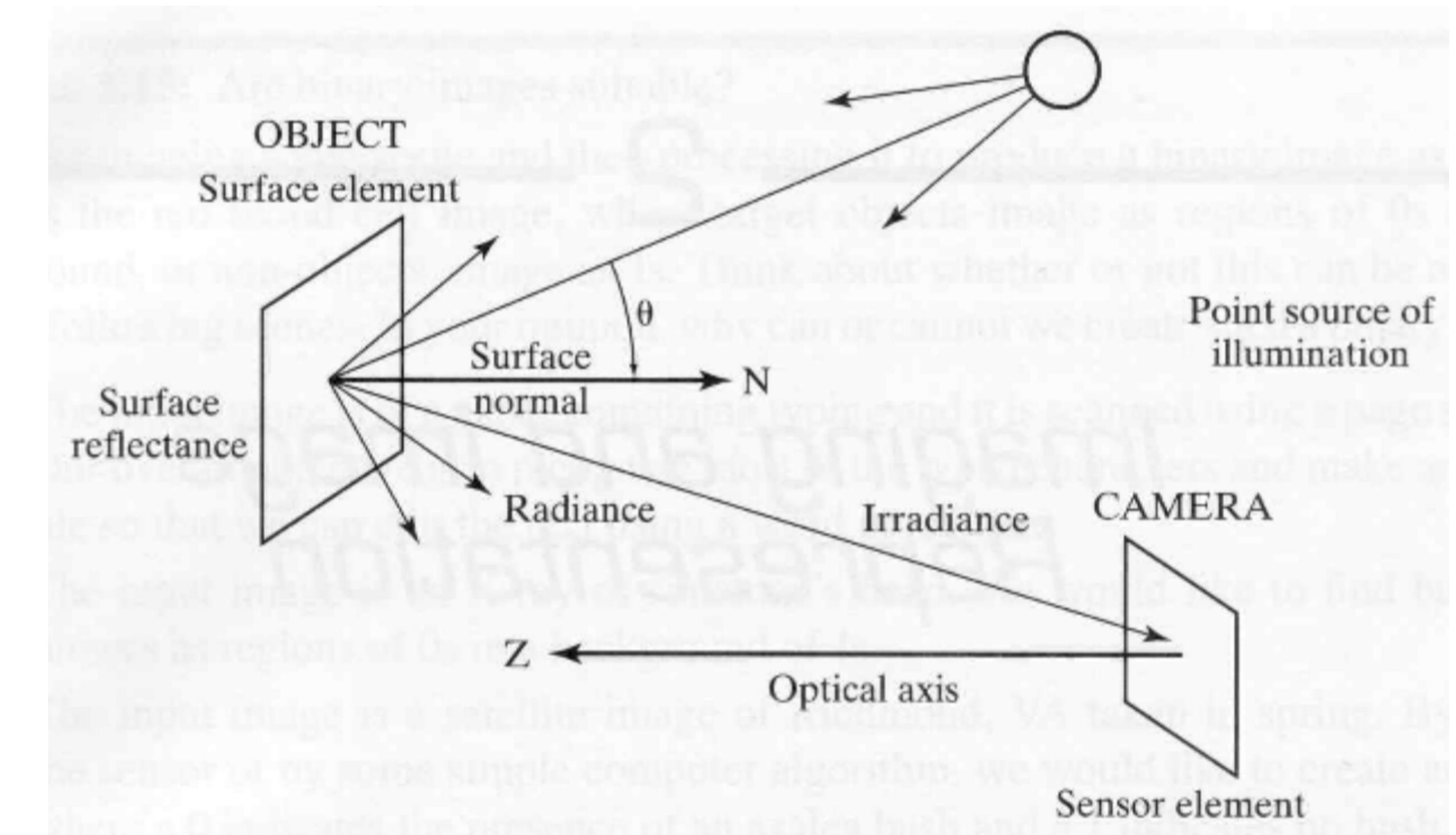


# Formación de la imagen

Compuesta de:

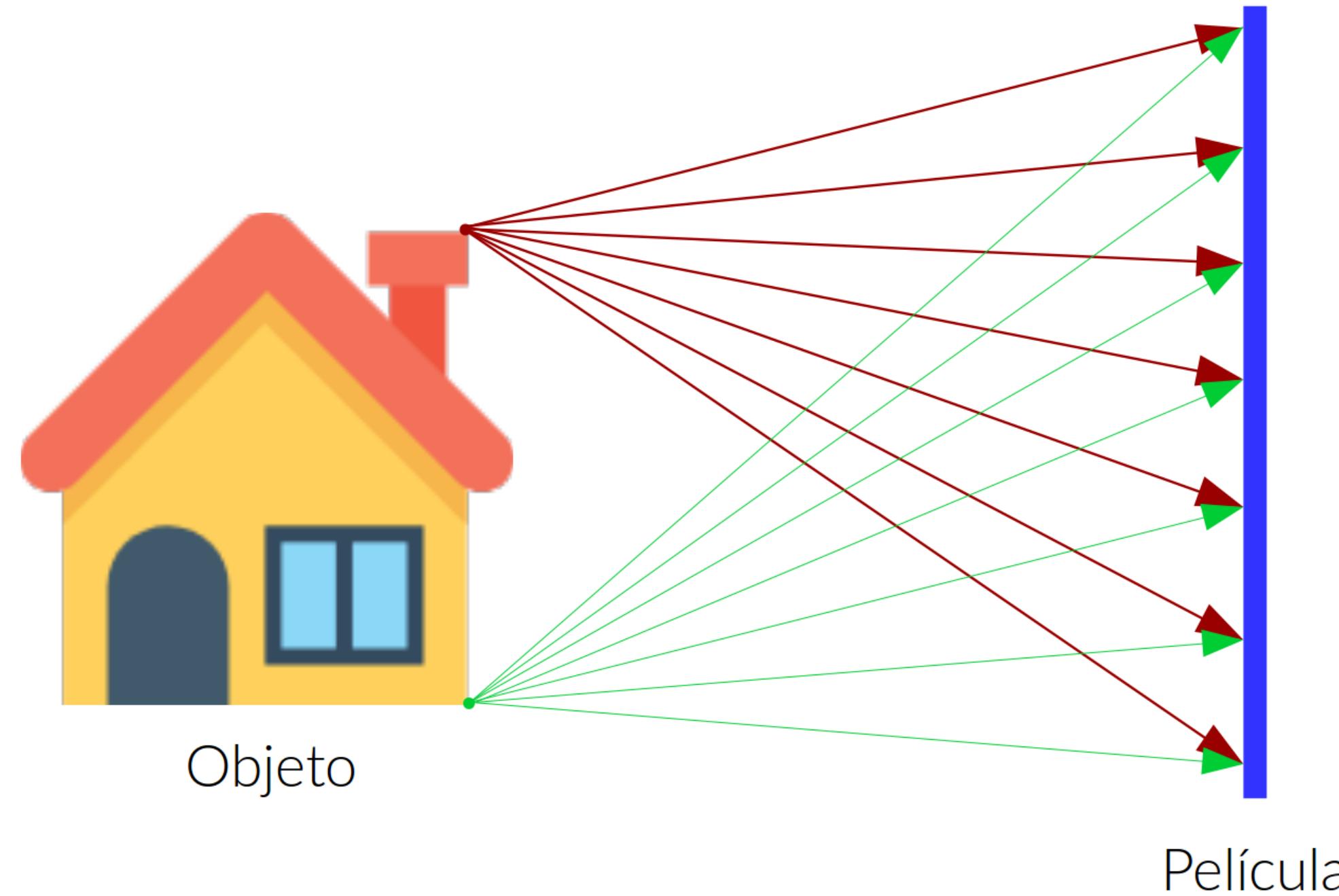
**1. Geométrica**  
correspondencia 3D - 2D

**2. Física de la luz** Brillo de  
cada punto de la imagen



**Modelo simplificado de la formación de imágenes. Tomado de Computer Vision - University of Nevada**

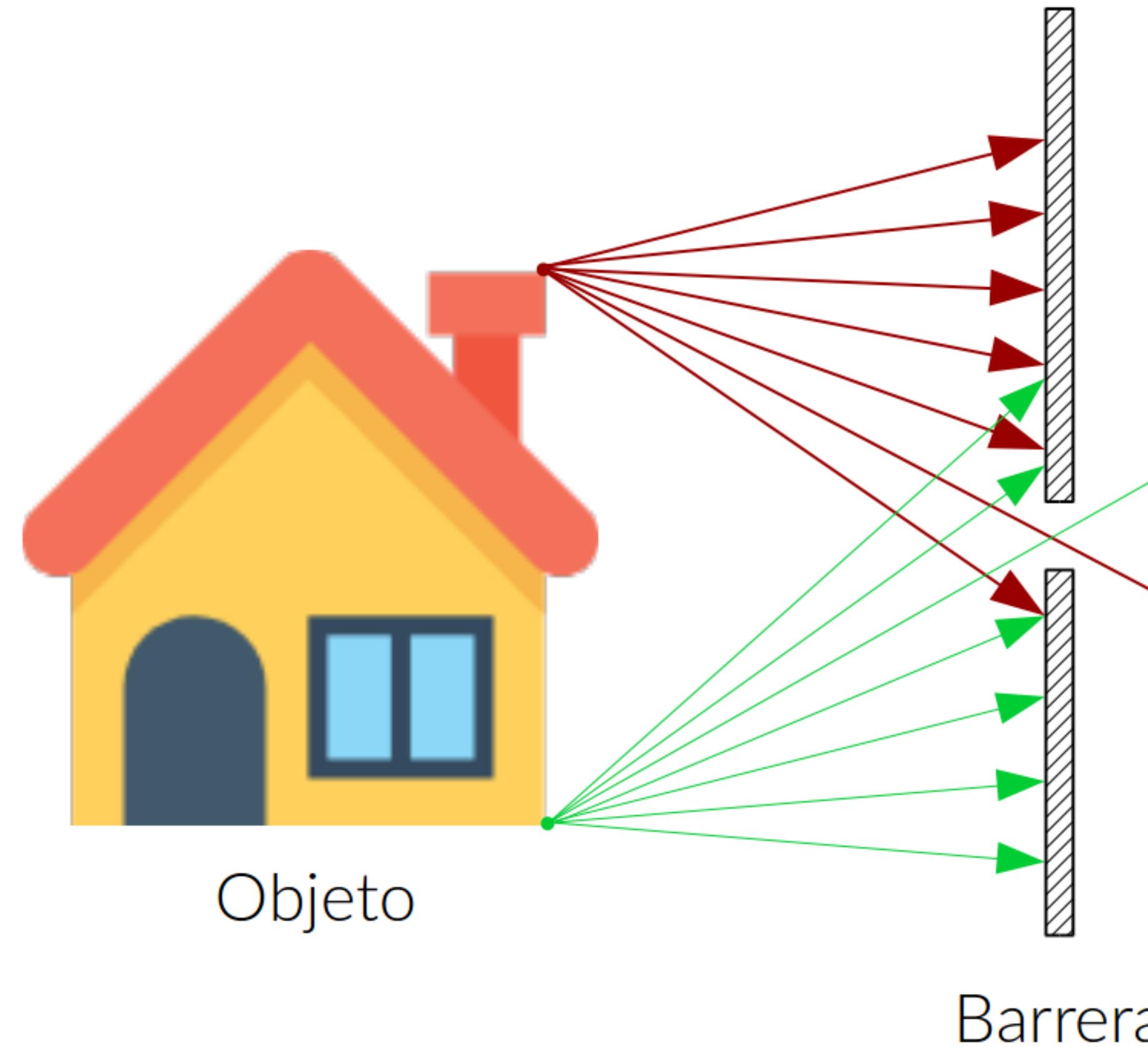
# Formación de la imagen



¿Si diseñamos una simple cámara?

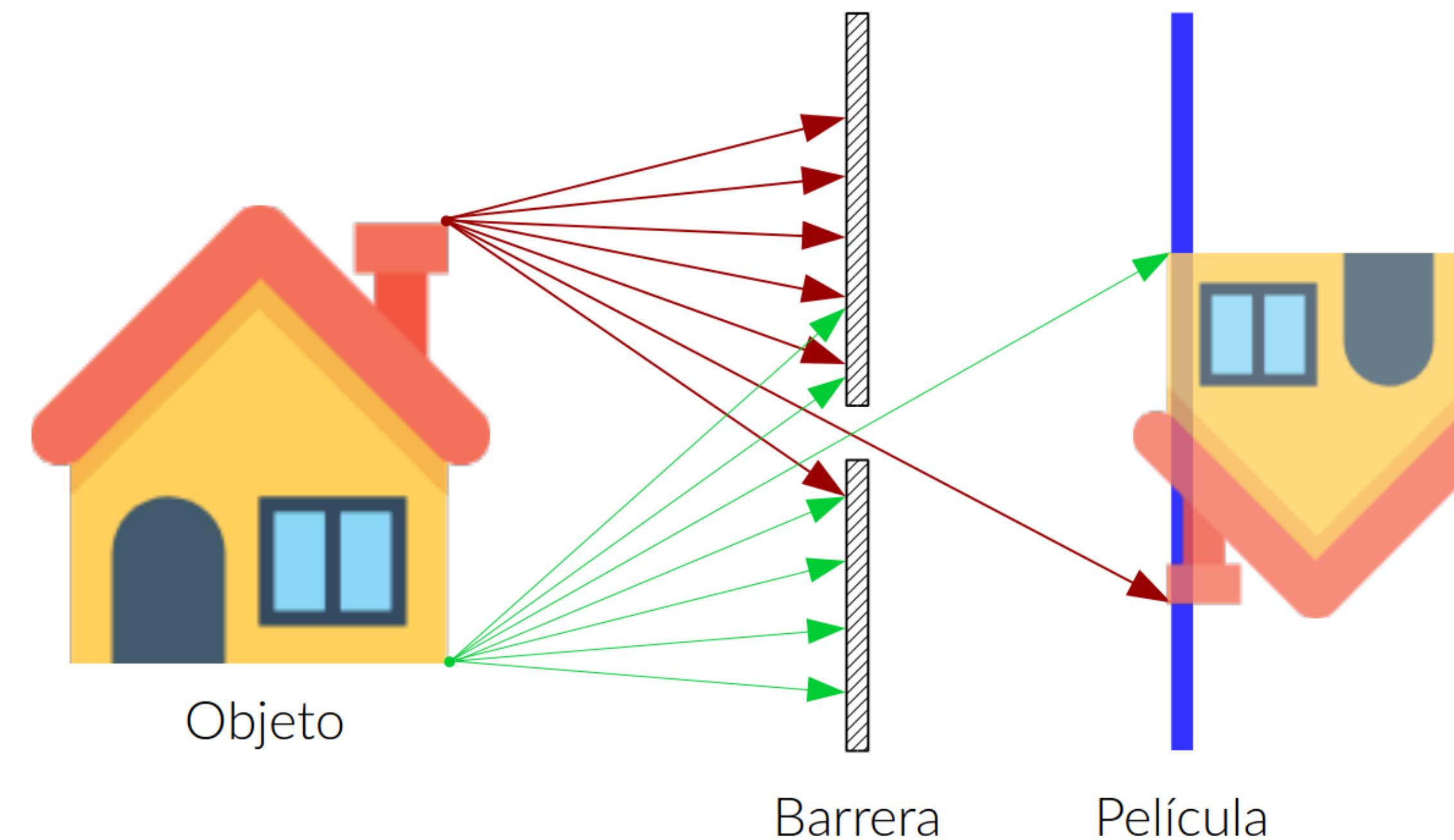
¿Cómo resultaría la imagen?

# Formación de la imagen



Agregamos una barrera para bloquear mayoría de rayos

# Formación de la imagen



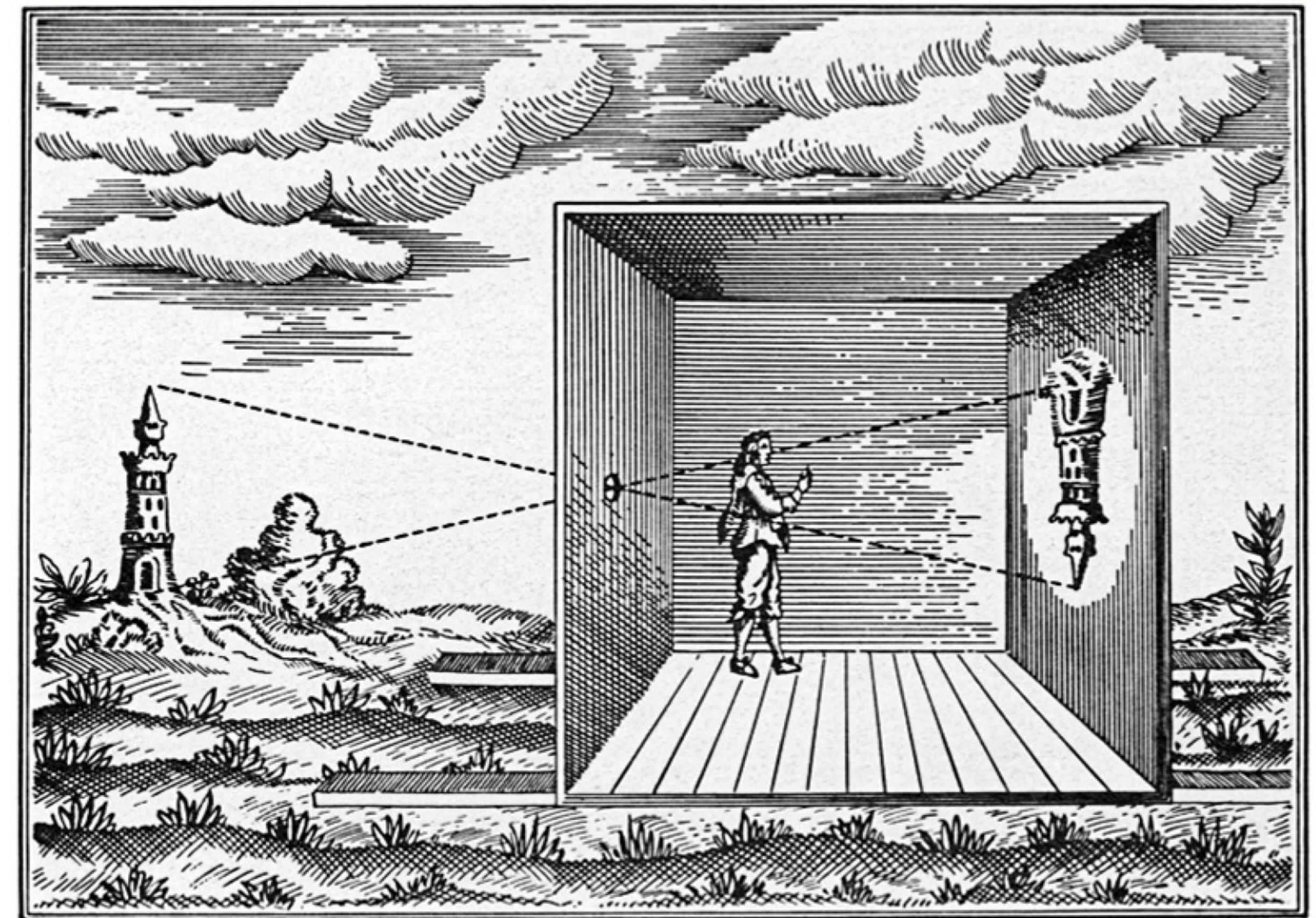
Agregamos una barrera para bloquear mayoría de rayos

Reducimos la borrosidad

¿Se deforma la imagen?

# Cámara Oscura

- Transferir una imagen compuesta por luz
- Pieza oscura y un agujero en una pared
- Estudio de perspectiva y eclipses
- No existían medios fotosensibles



*Tomado de photoion.co.uk*

# Cámara Oscura

## Usos

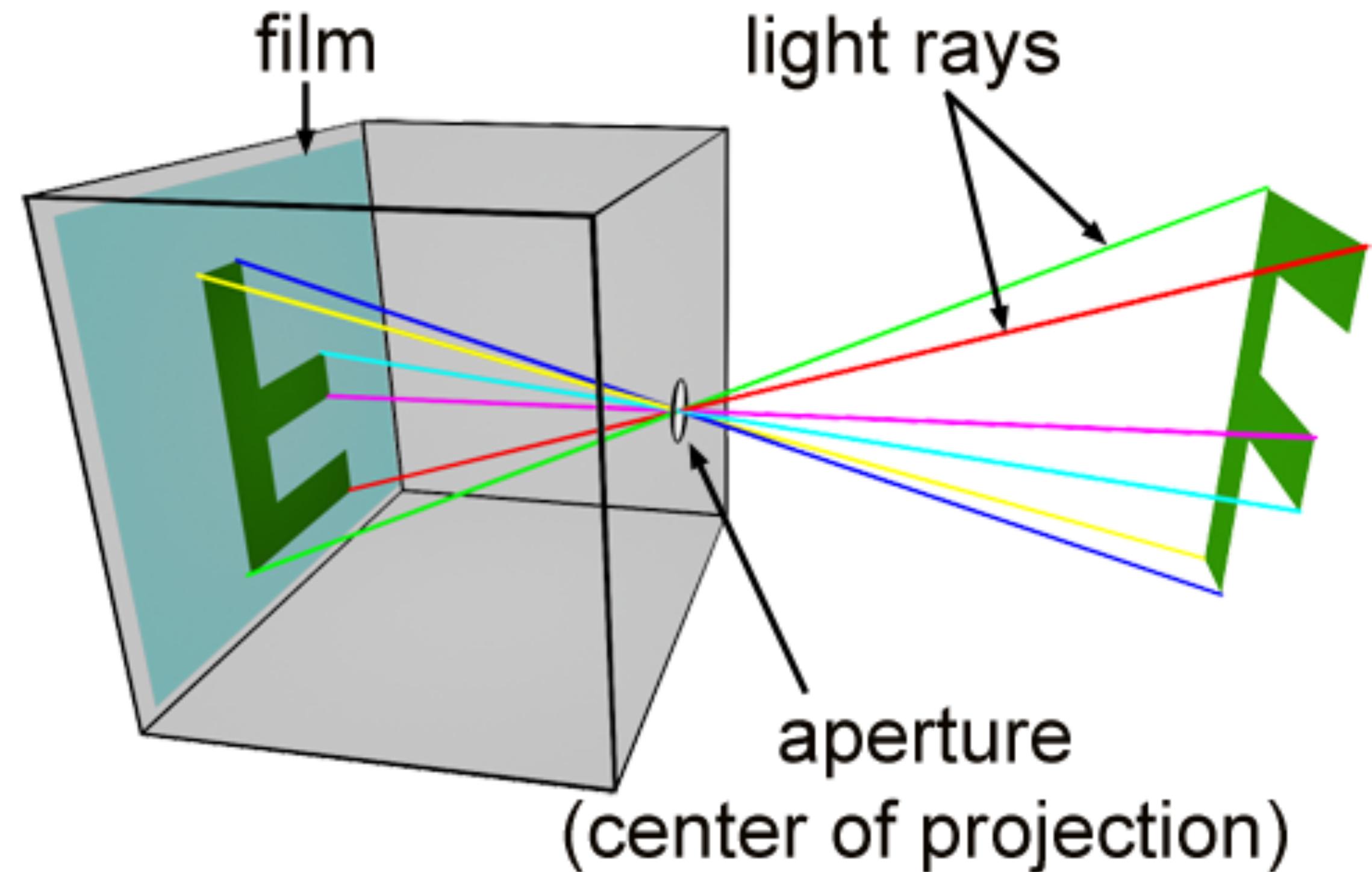
- Estudio de perspectiva
- Visualización de eclipses
- Ayuda para pintar



*Tomado de photoion.co.uk*

# Pinhole cámara

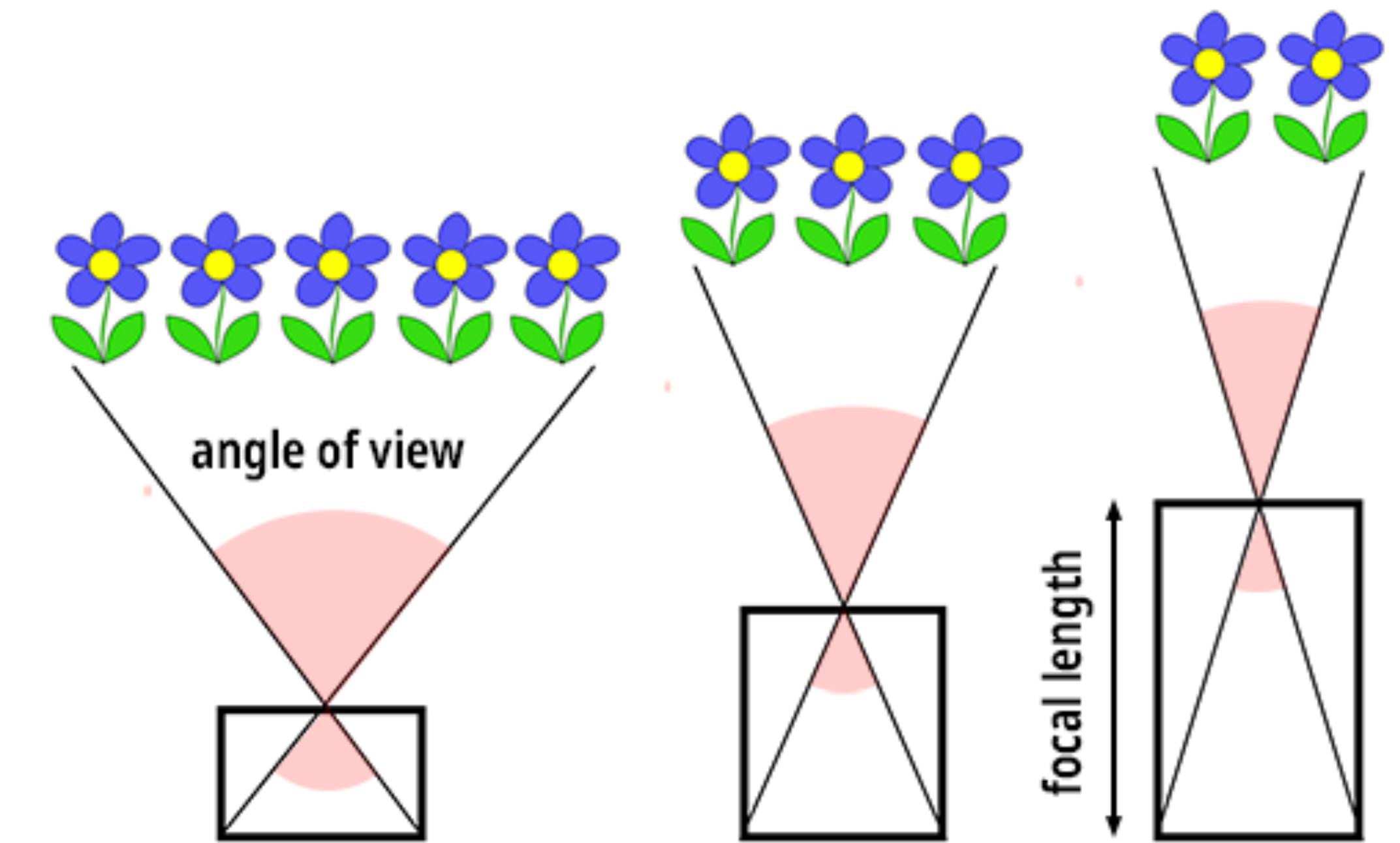
- Cámara oscura con un medio fotosensible
- Rayos pasan por el agujero
- Forma imagen rotada 180° con respecto al eje óptico



# Parámetros Cámara Pinhole

## “Distancia focal” (Pinhole)

- Distancia entre la apertura y la imagen del plano.
- Define el *ángulo de visión*

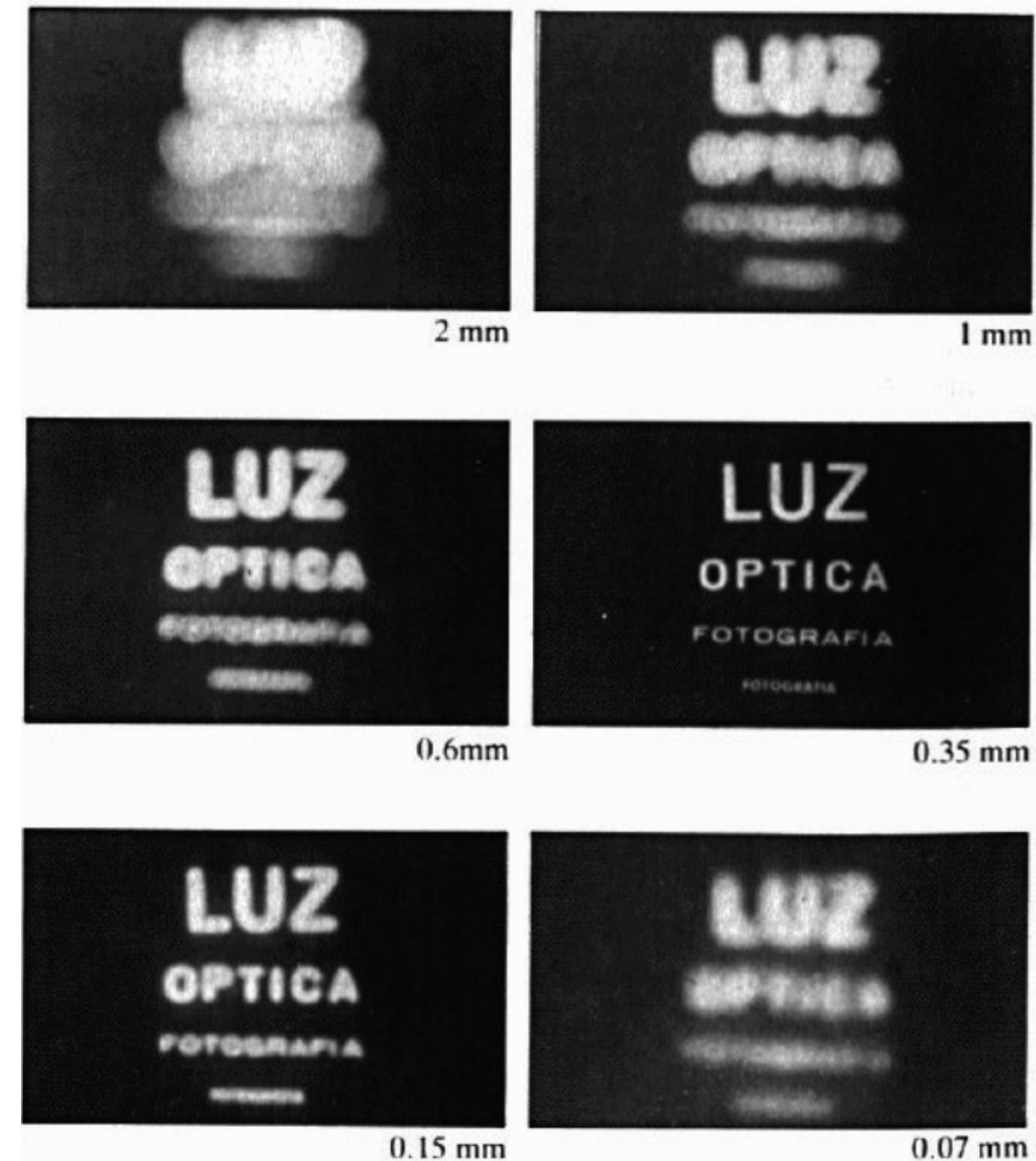


Tomado de scratchapixel.com

# Parámetros de cámara Pinhole

## Diámetro de la apertura

- Define que tan agudo y brillante es la imagen.
- Muy pequeño, muy poca luz ingresa y es sujeto a efectos de difracción

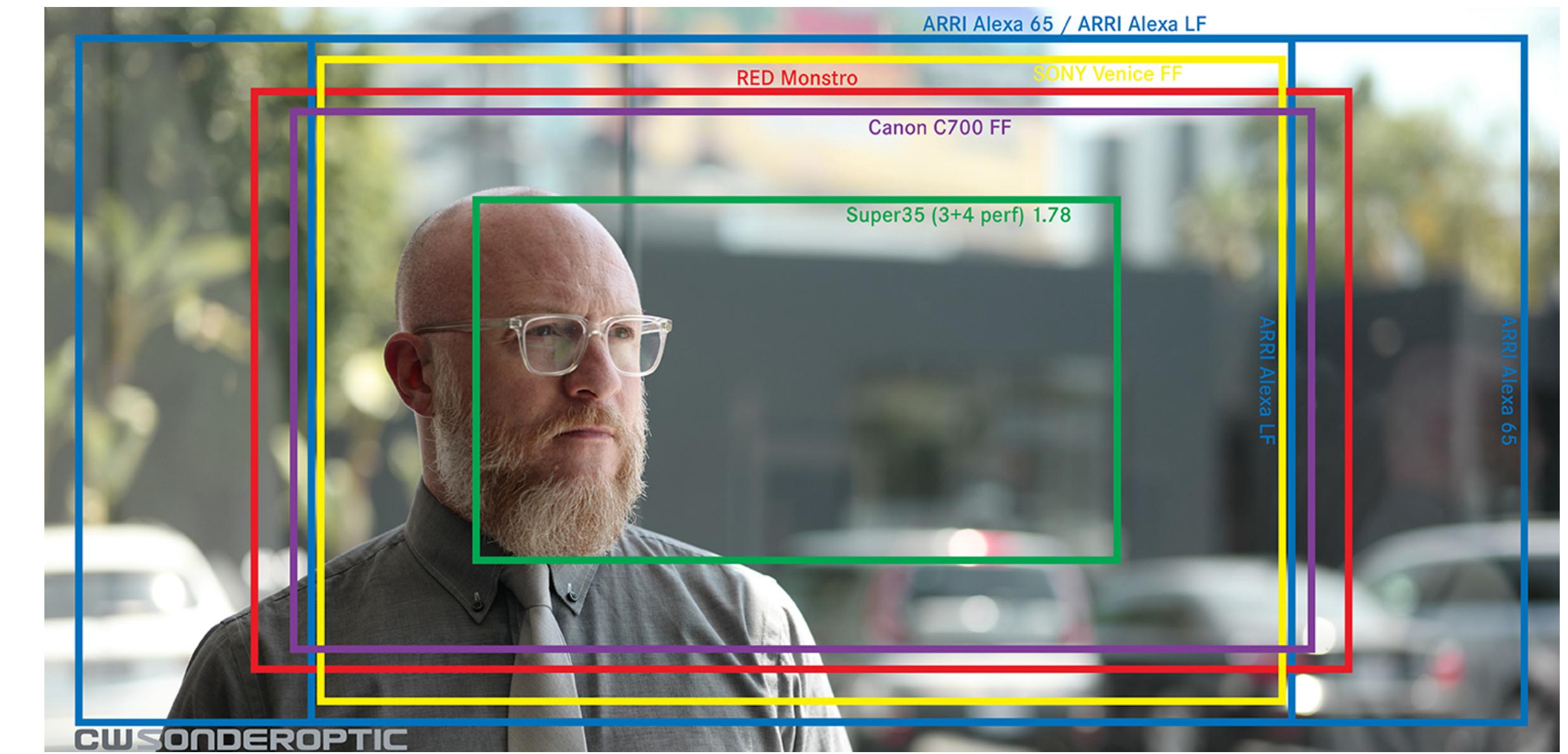


*Tomado de courses.cs.washington.edu*

# Parámetros de cámara Pinhole

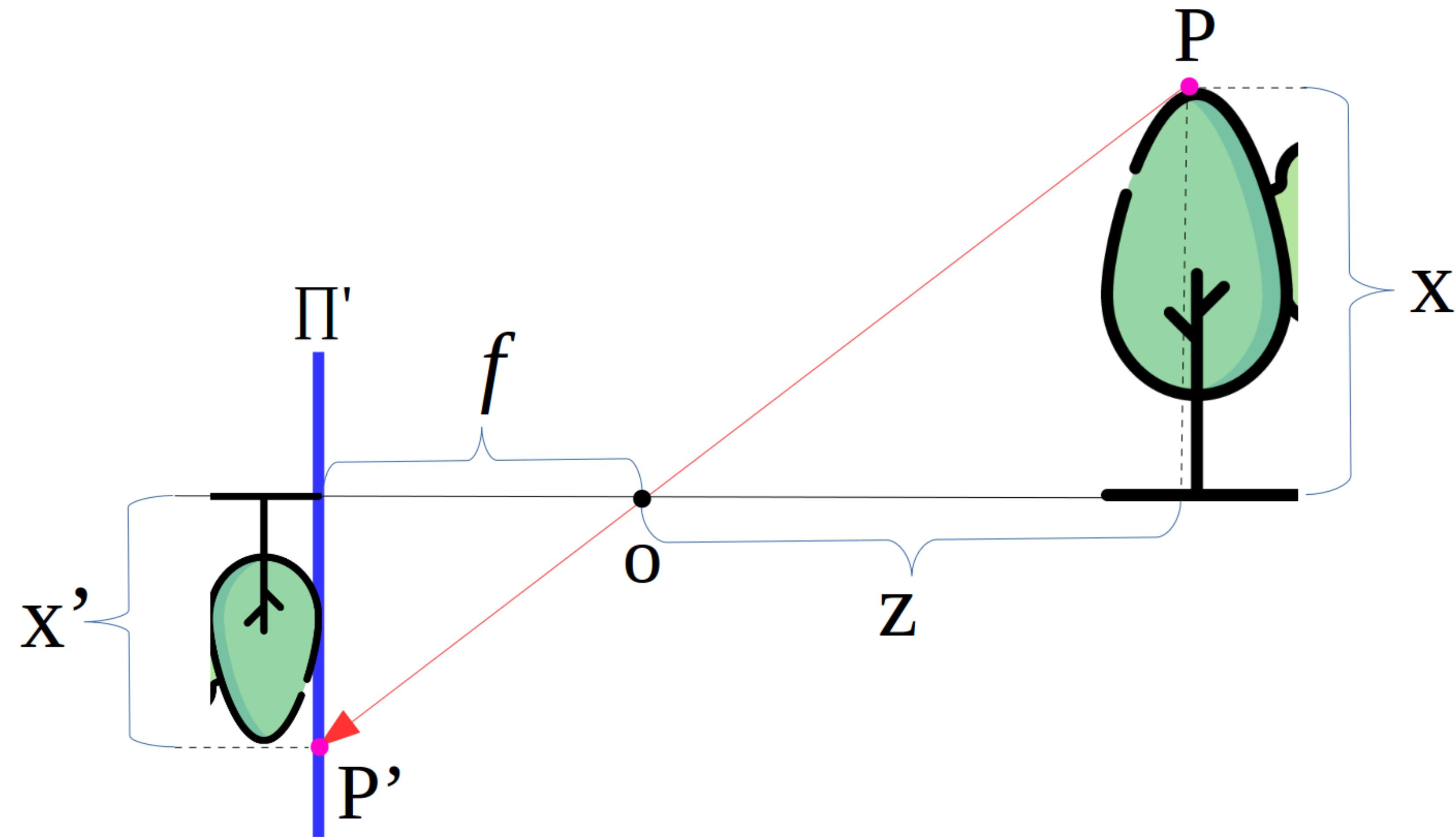
## Tamaño película o del sensor

- Define que tanta parte de la escena es capturada
- Este parámetro junto a la distancia focal controlan el ángulo de visión



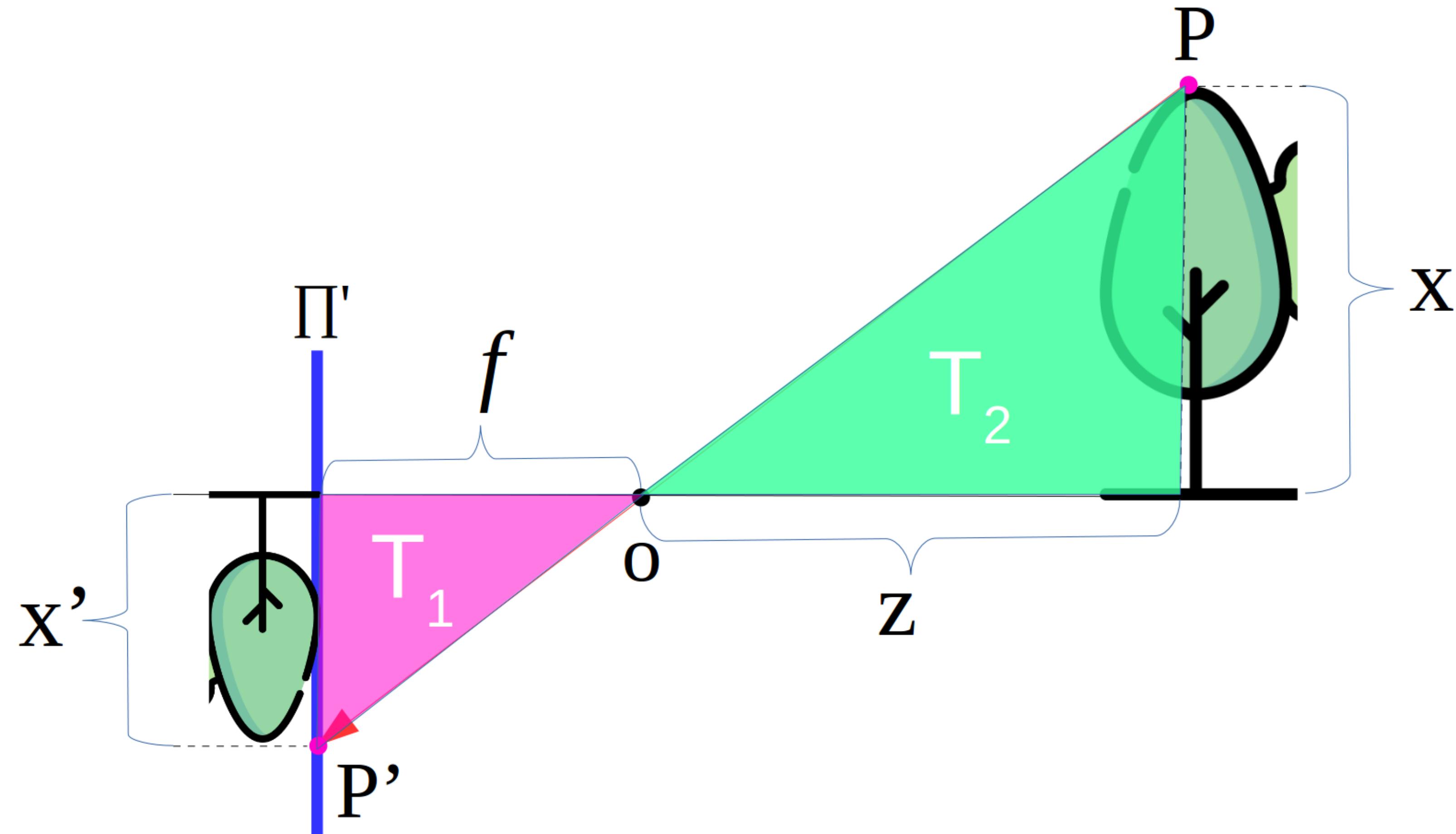
*Ángulo de visión con sensores de diferentes tamaños. Tomado de [ascmag.com](http://ascmag.com)*

# Modelo de cámara Pinhole 2D



Determinar la posición del punto  $P'$  en el plano de proyección  $\pi'$  del punto del objeto  $P$ , con apertura  $o$  y distancia focal  $f$

# Modelo de cámara Pinhole 2D



El triángulo  $\triangle T_1$  es similar al triángulo  $\triangle T_2$ , por lo que los ratios entre sus ángulos son iguales

# Modelo de cámara Pinhole 2D

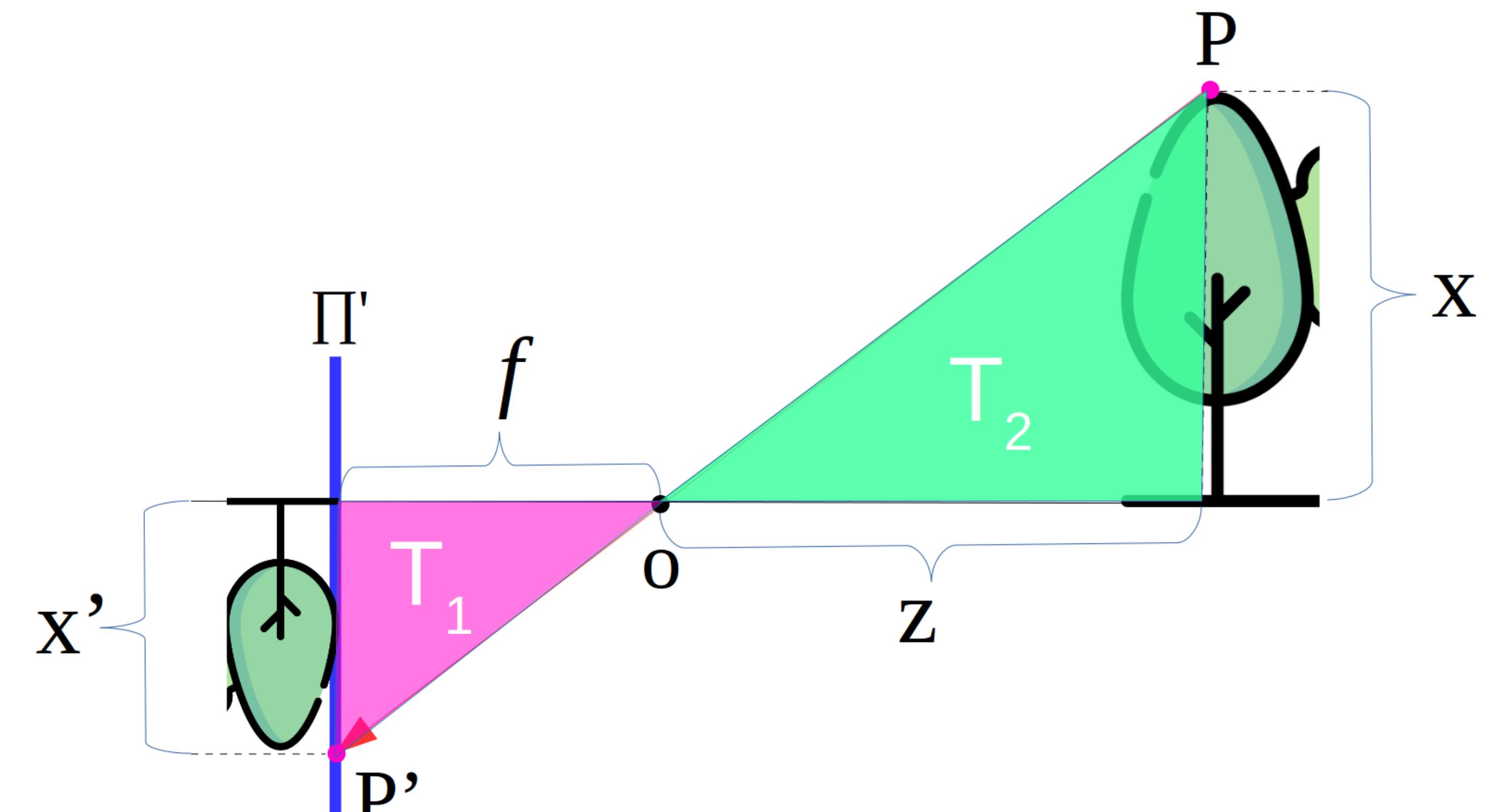
Por triángulos similares

$$\frac{x}{z} = \frac{x'}{f}$$

Organizando e igual en el eje y

$$x' = f \frac{x}{z}$$

$$y' = f \frac{y}{z}$$



# Modelo de cámara Pinhole 2D

Como operación matricial en coordenadas homogéneas:

$$\begin{bmatrix} x' \\ y' \\ z' \\ w \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/f & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ z/f \end{bmatrix}$$

Luego al pasar a cartesianas

$$[f\frac{x}{z}, f\frac{y}{z}, f]^T$$

**Coordenadas homogéneas**

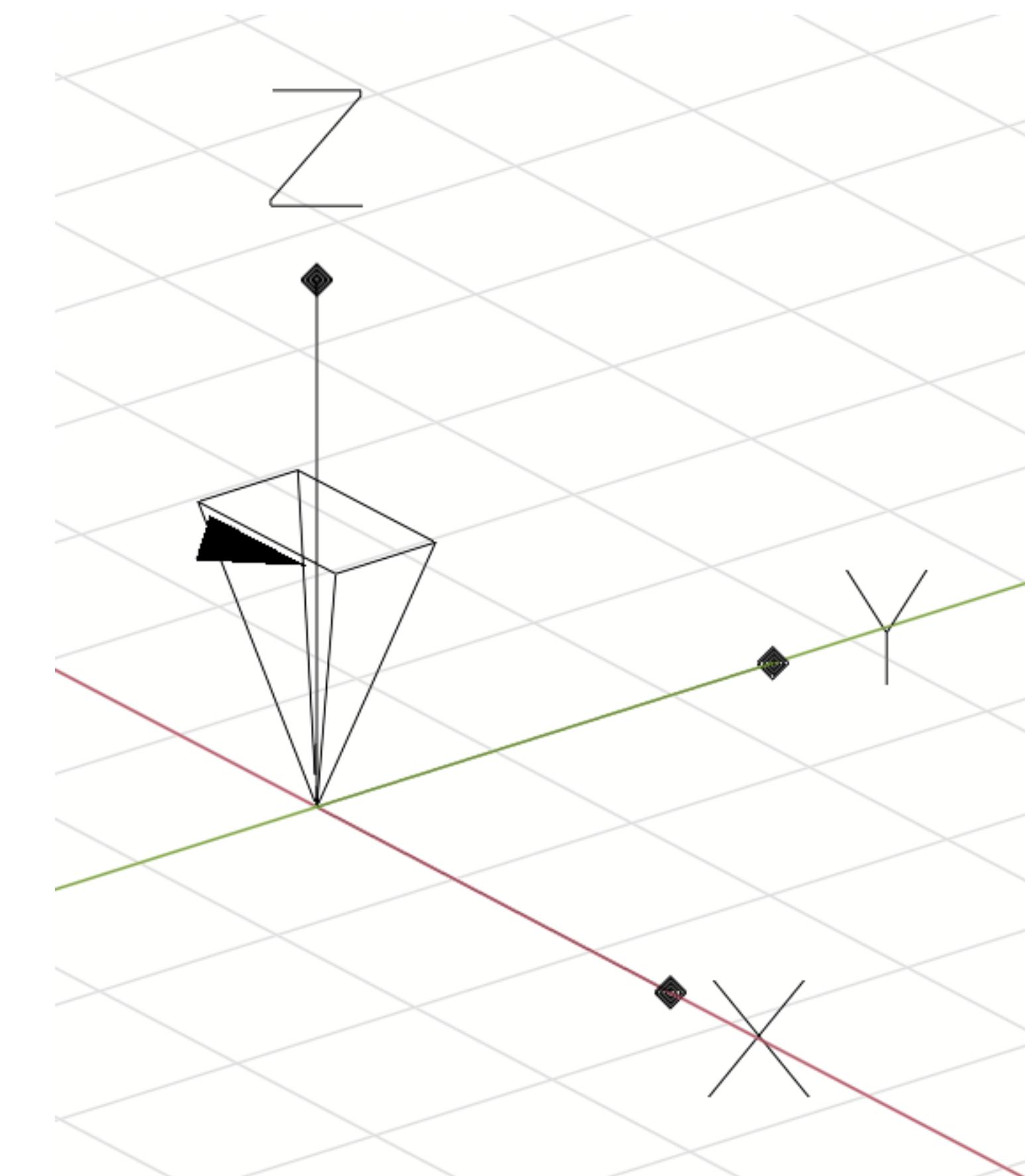
<https://www.tomdalling.com/blog/modern-opengl/explaining-homogenous-coordinates-and-projective-geometry/>

# Modelo de cámara Pinhole 2D

## Posición de la cámara

Según las operaciones anteriores la cámara se encuentra:

- Centrada en el origen
- Alineada al eje z



Posición por defecto de la cámara virtual

# Modelo de cámara Pinhole 2D

## Conclusiones

- Simula el comportamiento ideal de una cámara Pinhole
- No tiene en cuenta distorsiones de lentes o de la apertura
- Efectos son pequeños y pueden ser compensados
- Normalmente usada en CG



*Architectural Rendering Exterior por Desert Rose*

# Lentes

- Perforación reemplazada por los lentes
- Mitigan el problema entre definición vs. brillo

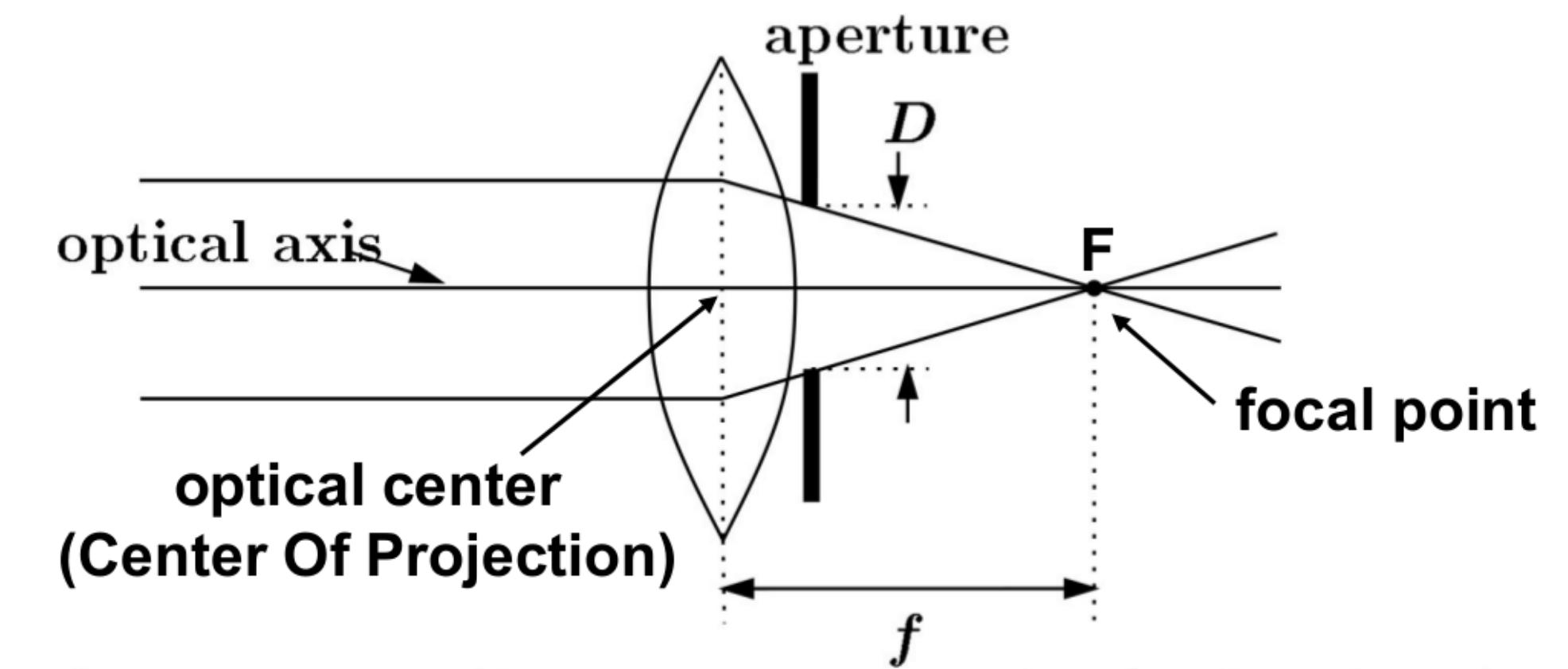
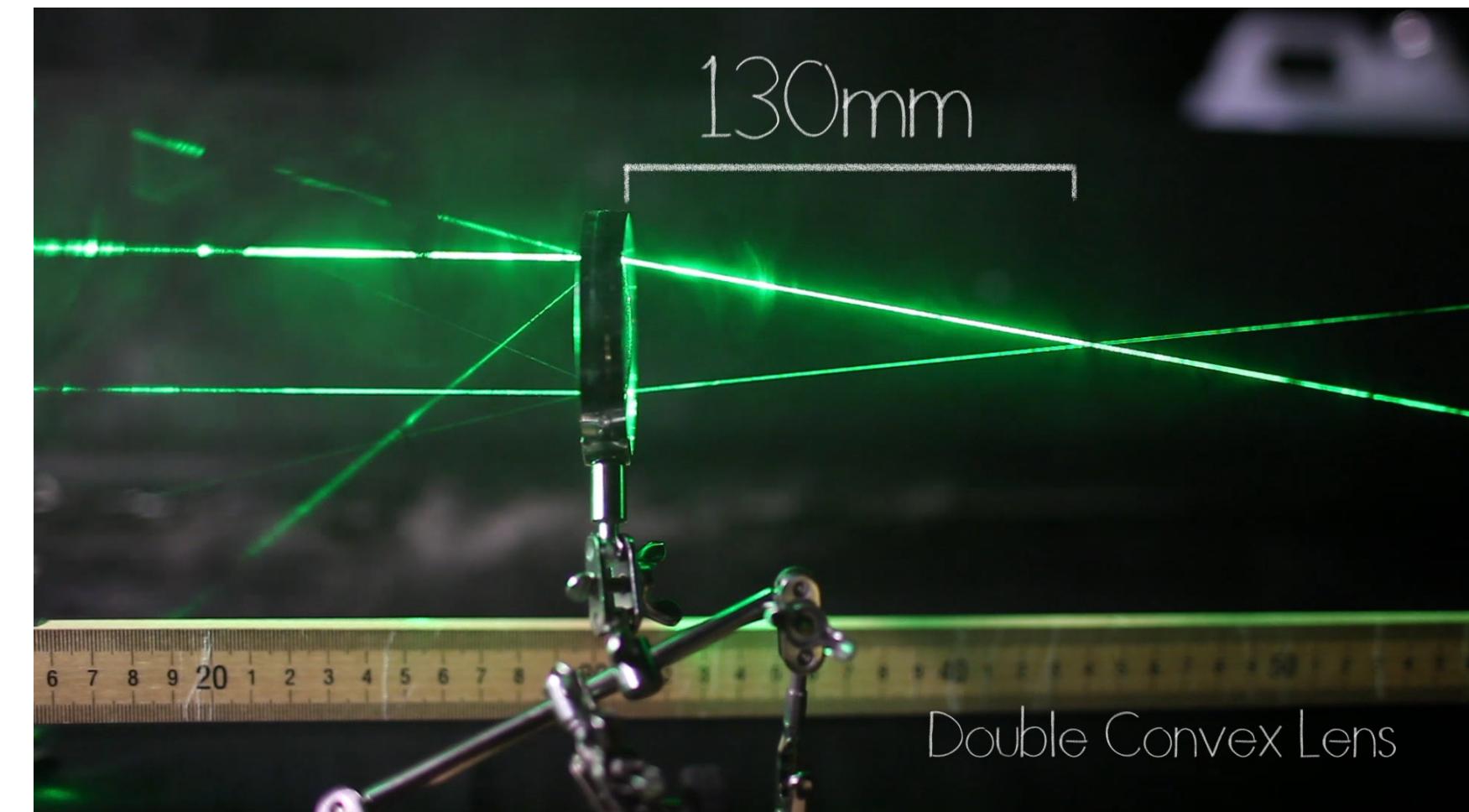
*Tamaño y posición correcta satisface la propiedad de que todo punto emitido de la escena converge a un único punto*



*Por Daven Mathies en Digital Trends*

# Características Cámaras de lente

- Rayos que pasan por el centro no son desviados
- Más lejos del centro más desviación
- Rayos paralelos convergen en un mismo punto
- El punto en el que convergen los rayos paralelos eje óptico es llamado Punto Focal



# Distorsiones de los lentes



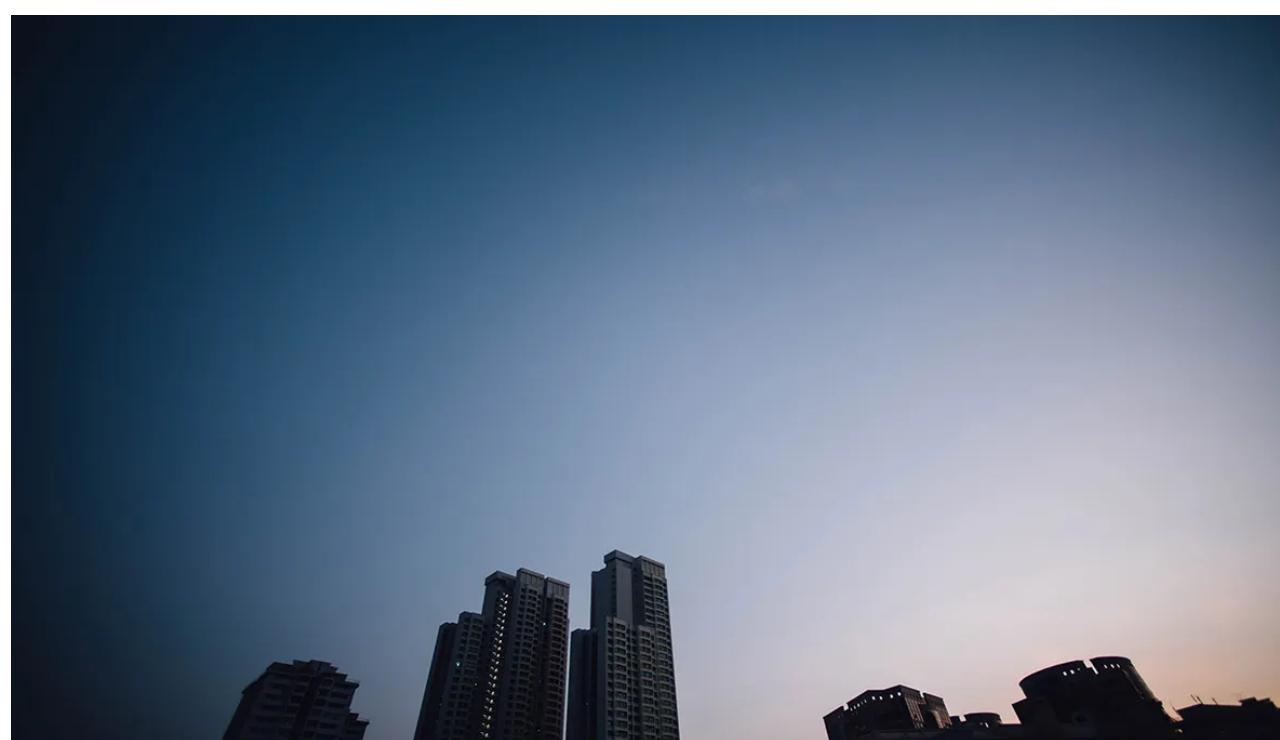
Distorsión radial



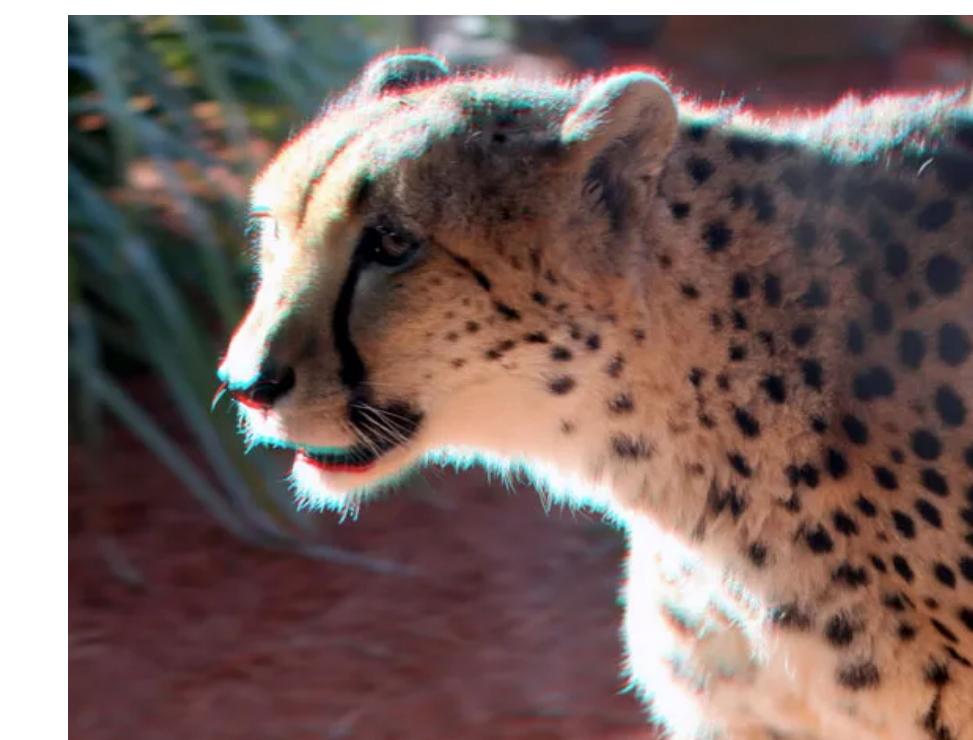
Enfoque



Exposición



Vignetting



Chromatic aberration

# Ejercicio 1

- Implementar las siguientes funciones de transformaciones 2D
  - Escalar
  - Rotar
  - Transladar
- Las funciones reciben la información de una malla en coordenadas homogéneas en donde cada vértice es una fila de una matriz



# Manejo de imágenes

## Imágenes como funciones

Pensar la imagen es una función. Cada pixel retorna la intensidad

$$f(x, y) \rightarrow I$$

Imágenes a color

$$f(x, y) = \begin{bmatrix} r(x, y) \\ g(x, y) \\ b(x, y) \end{bmatrix}$$

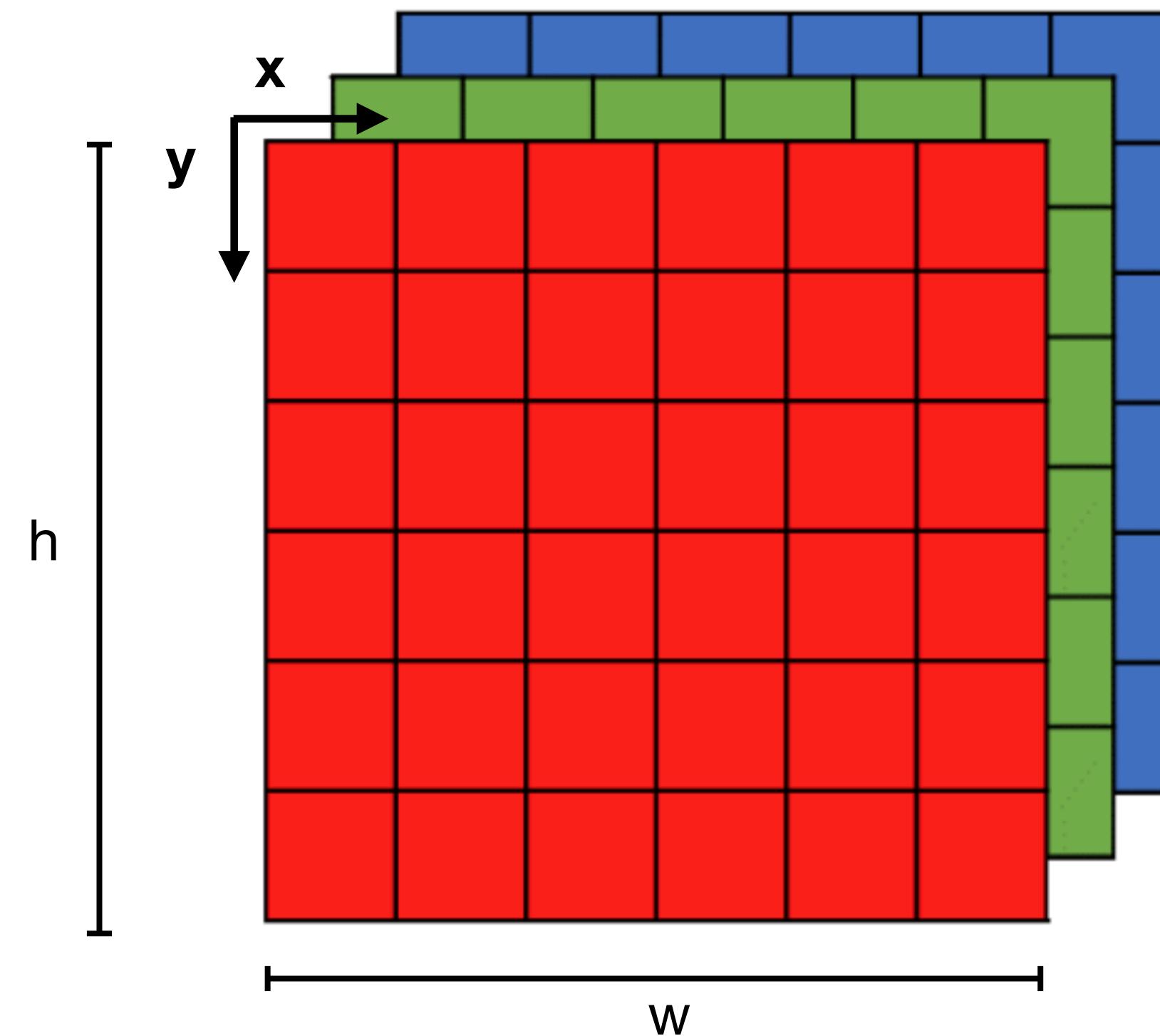
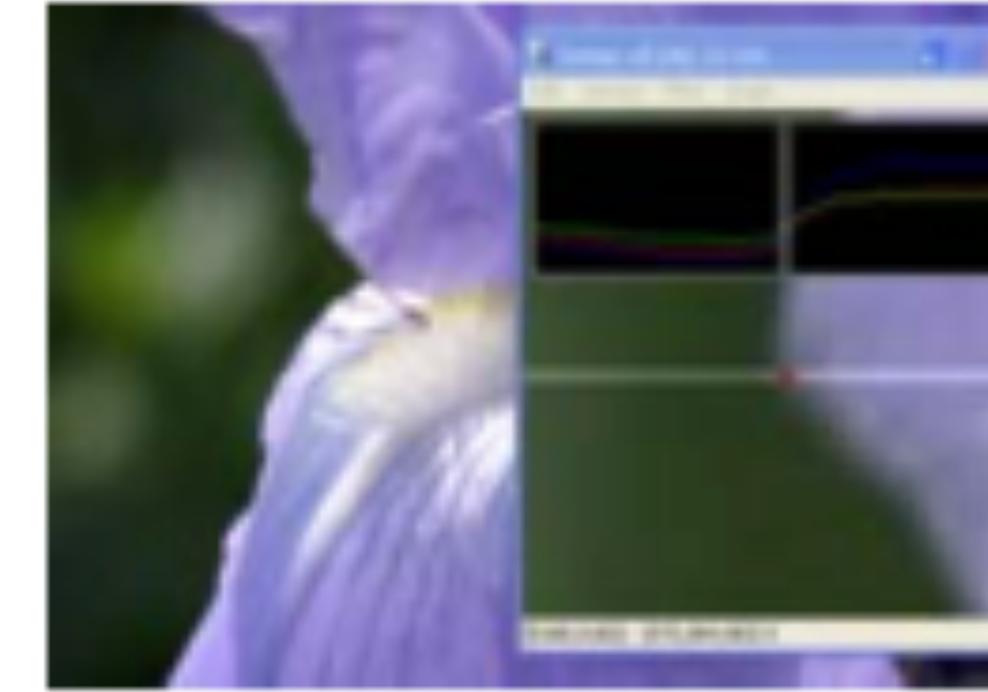


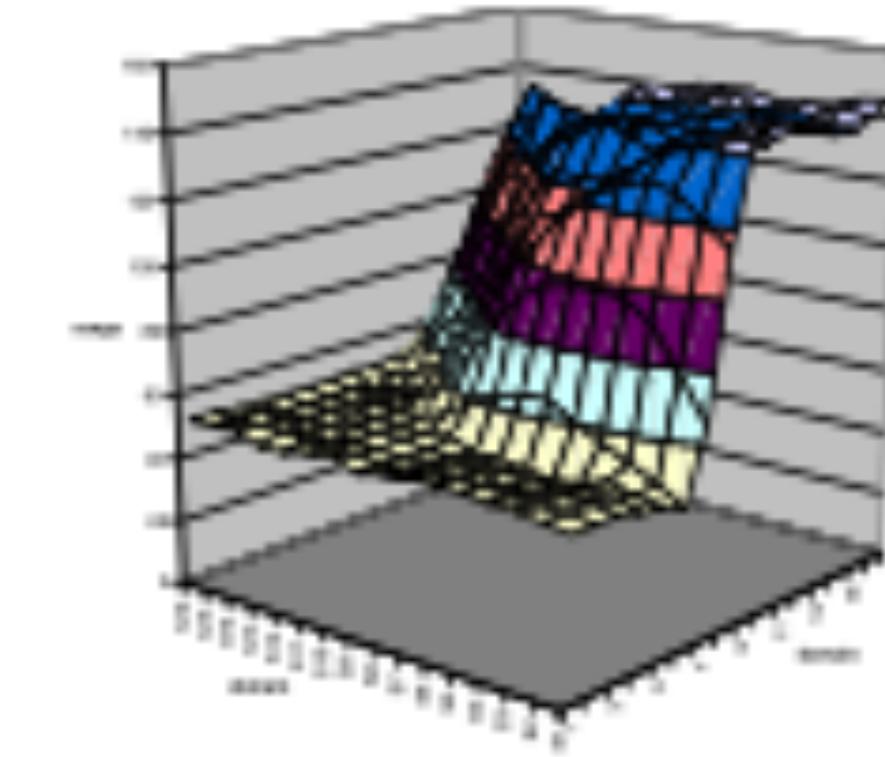
Imagen digital.  $n = w * h * 3$

# Manejo de imágenes

Representación de una imagen



45	60	98	127	132	133	137	133
46	65	98	123	126	128	131	133
47	65	96	115	119	123	135	137
47	63	91	107	113	122	138	134
50	59	80	97	100	103	133	134
49	53	68	85	97	113	128	133
50	50	58	70	84	102	116	126
50	50	52	58	69	86	101	120



- Por su color (apariencia),
- Cuadrícula de números
- Función bidimensional (gráfico de superficie).

# Procesamiento de imágenes

- Uso del procesamiento de imágenes para procesar la imagen y convertirla en una forma adecuada
- Asignar valores de píxeles de una imagen a otra
- Tipo más sencillo de transformaciones de imagen, manipulan cada píxel independientemente de sus vecinos.
- Estas transformaciones se denominan *point operations*



Algunas operaciones de tratamiento de imágenes: (a) imagen original; (b) aumento del contraste; (c) cambio del tono; (d) "posterizada" (colores cuantizados); (e) difuminada; (f) girada.

# Operadores de punto

- El valor de cada píxel de salida depende del valor del píxel de entrada
- Operador que toma una o más imágenes de entrada y produce una imagen de salida

$$g(i, j) = h(f(i, j))$$

- Ejemplos:
  - Corrección de la exposición
  - Balance de blancos (Color balance)
  - Reducción del ruido de la imagen
  - Aumento de la nitidez
  - Transformaciones geométricas



**Balance de blancos (Color balance)**

# Brillo y contraste

- Multiplicación y la suma de una constante.

$$g(x) = af(x) + b$$

- Los parámetros  $a > 0$  and  $b$  son llamados **gain** y **bias**
- Se dice que estos parámetros controlan el contraste y el brillo



Inicial

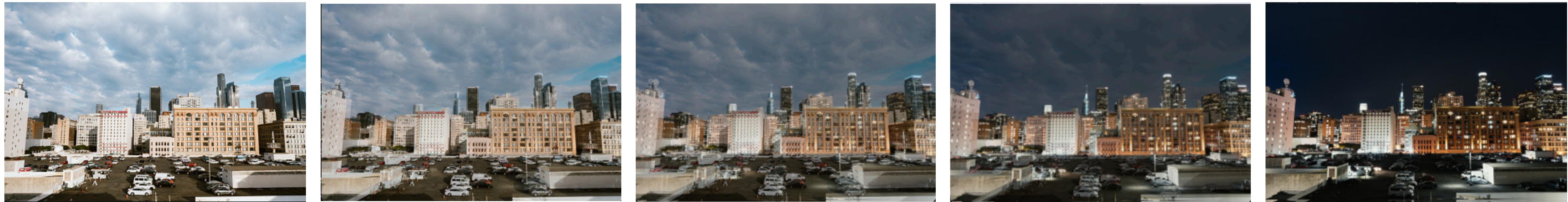


$\alpha = 1.5; \beta = 0.2$

# Mezcla lineal

Realizar una disolución cruzada temporal entre dos imágenes o vídeos

$$g(x) = (1 - \alpha)f_0(x) + \alpha f_1(x)$$



0

0.25

0.5

0.75

1

# Corrección gamma

- Traducción entre la sensibilidad de nuestros ojos y los sensores de una cámara fotográfica
- En una cámara digital, cuando el sensor recibe el doble de fotones, recibe el doble de señal
- Percibimos el doble de luz como si fuera sólo una fracción más brillante
- Operación:

$$g(x) = f(x)^\gamma$$



Inicial



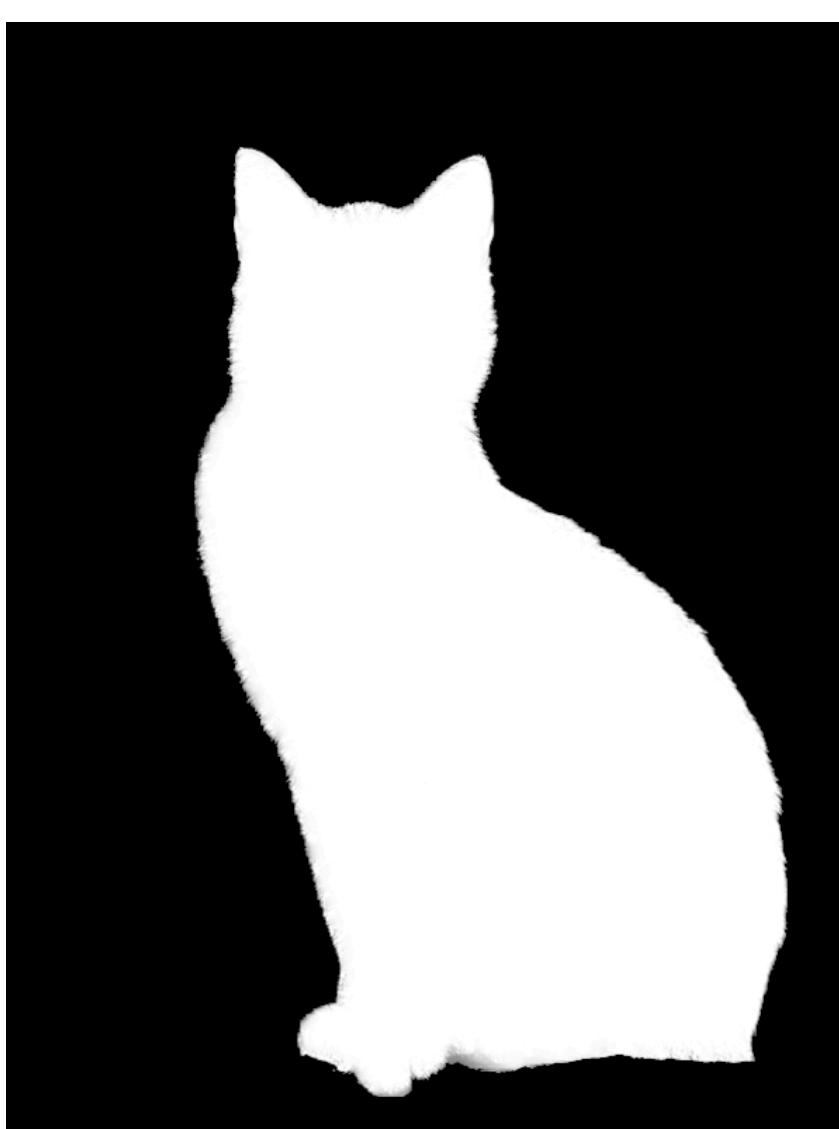
$$\gamma = 2$$

# Image matting

- Recortar un objeto en primer plano de una escena y colocarlo sobre un fondo diferente.
- El proceso de extracción del objeto de la imagen original suele denominarse “matting”
- Se usa una imagen intermedia llamada *alpha-matted*
- Además de los tres canales de color RGB, una imagen con matiz alfa contiene un cuarto canal alfa (o A) que describe la cantidad relativa de opacidad



Inicial F



Alpha  $\alpha$



Compuesta C

Compositing equation:

$$C = (1 - \alpha)B + \alpha F$$

# Ejercicio 2

1. Crear un script que cambie el color de fondo de la imagen
2. Implementar los operadores de punto y realizar un ejemplo con cada uno



Imagen inicial



# Ejercicio 2

## Canales de las imágenes

```
# El orden de los canales es: BGR  
im = cv2.imread('ruta/de/la/imagen')  
green = im[:, :, 1]
```



## Boolean indexing

```
a = np.array(range(30))
```

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  
     8,  9, 10, 11, 12, 13, 14, 15, 16, 17,  
    18, 19, 20, 21, 22, 23, 24, 25, 26, 27,  
    28, 29])
```

```
a[a>10] = 0  
array([ 0,  1,  2,  3,  4,  5,  6,  7,  
     8,  9, 10,  0,  0,  0,  0,  0,  0,  
     0,  0,  0,  0,  0,  0,  0,  0,  0,  
     0,  0])
```