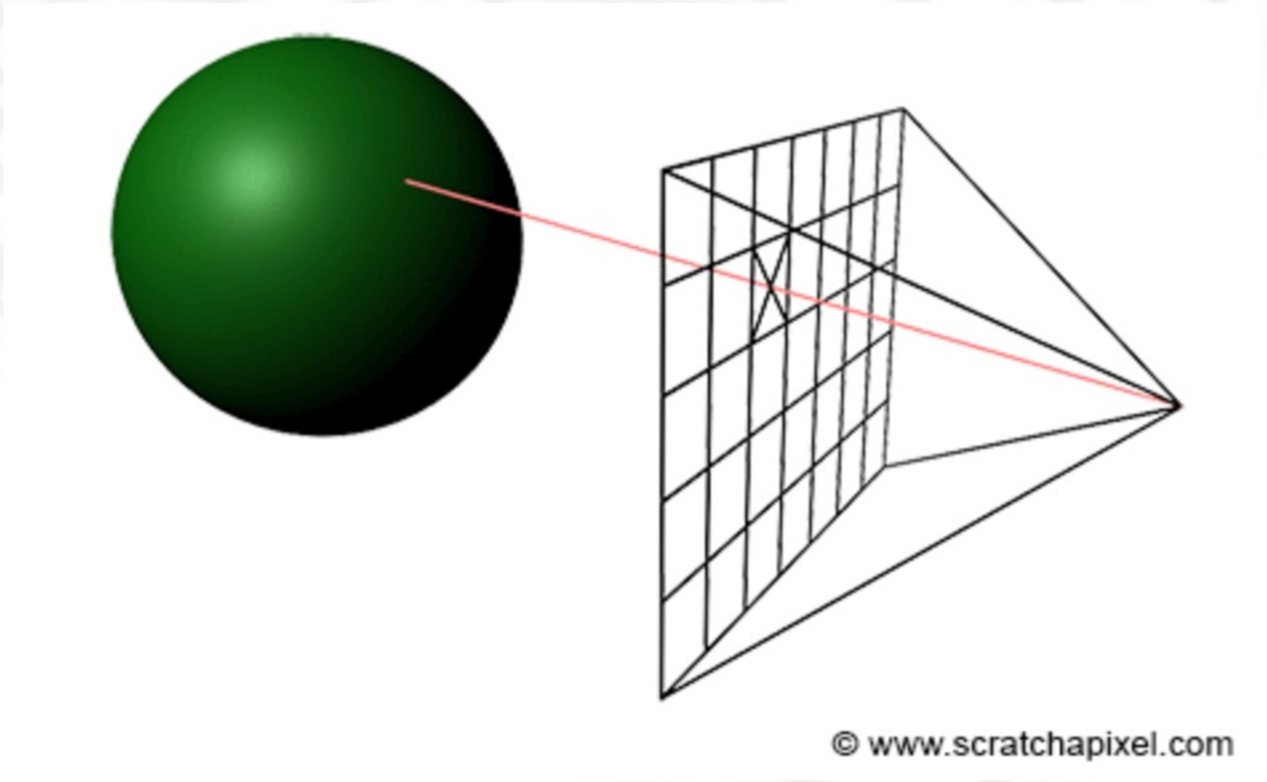
Mestrado em
Engenharia Informática

VI-RT
Perspective Camera
Image

Visualização e Iluminação

PERSPECTIVE CAMERA



Raster -> Screen -> Camera Space

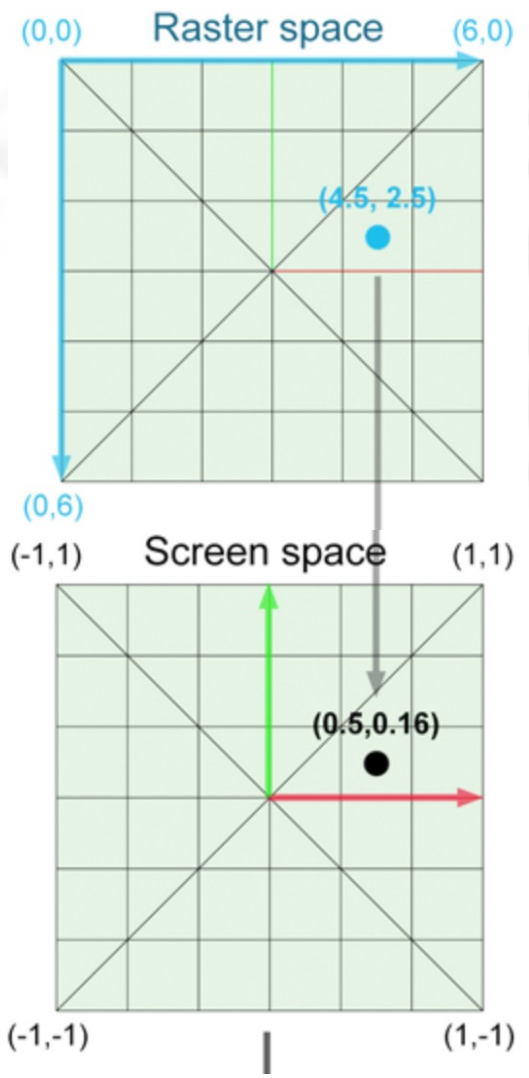
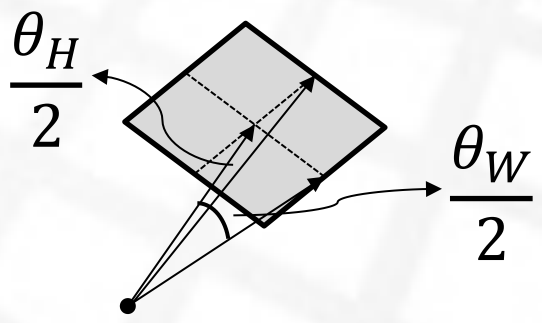


Image Resolution = (W,H)

$$x_s = \frac{2(x+0.5)}{W} - 1 \qquad y_s = \frac{2((H-y-1)+0.5)}{H} - 1$$

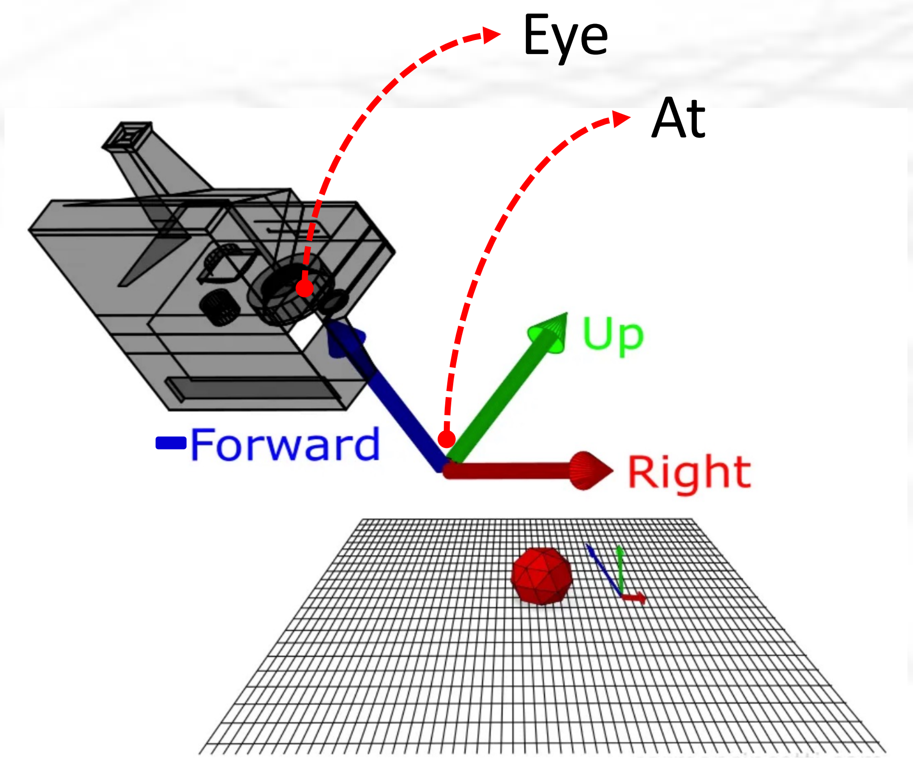


$$x_c = x_s * \tan\frac{\theta_W}{2}$$

$$y_c = y_s * \tan\frac{\theta_H}{2}$$

$$\begin{pmatrix} x_c \\ y_c \\ 1 \end{pmatrix}$$

Camera Setup



$$F = normalize(At - Eye)$$

$$R = normalize(cross(F, U))$$

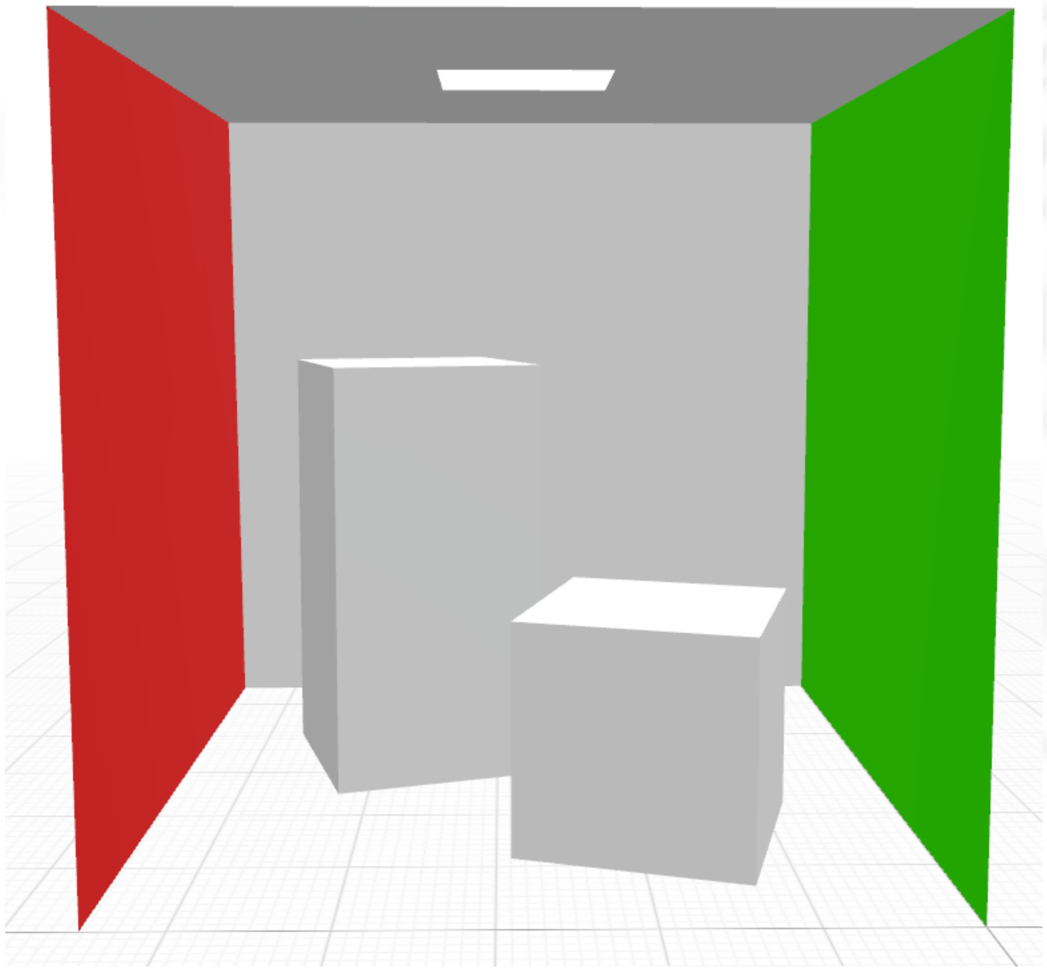
$$U = normalize(cross(R, F))$$

$$c2w = \begin{bmatrix} R_x & R_y & R_z \\ U_x & U_y & U_z \\ F_x & F_y & F_z \end{bmatrix}$$

$$ray.dir = \begin{bmatrix} R_x & R_y & R_z \\ U_x & U_y & U_z \\ F_x & F_y & F_z \end{bmatrix} \begin{pmatrix} x_c \\ y_c \\ 1 \end{pmatrix}$$

$$ray.o = eye$$

cornell_box.obj



floor ~(0,0,0) → (560,0,560)

ceiling ~(0,550,0) → (560,550,560)

back ~(0,0,560) → (560,550,560)

front ~(0,0,0) → (560,550,0)

green ~(0,0,0) → (0,550,560)

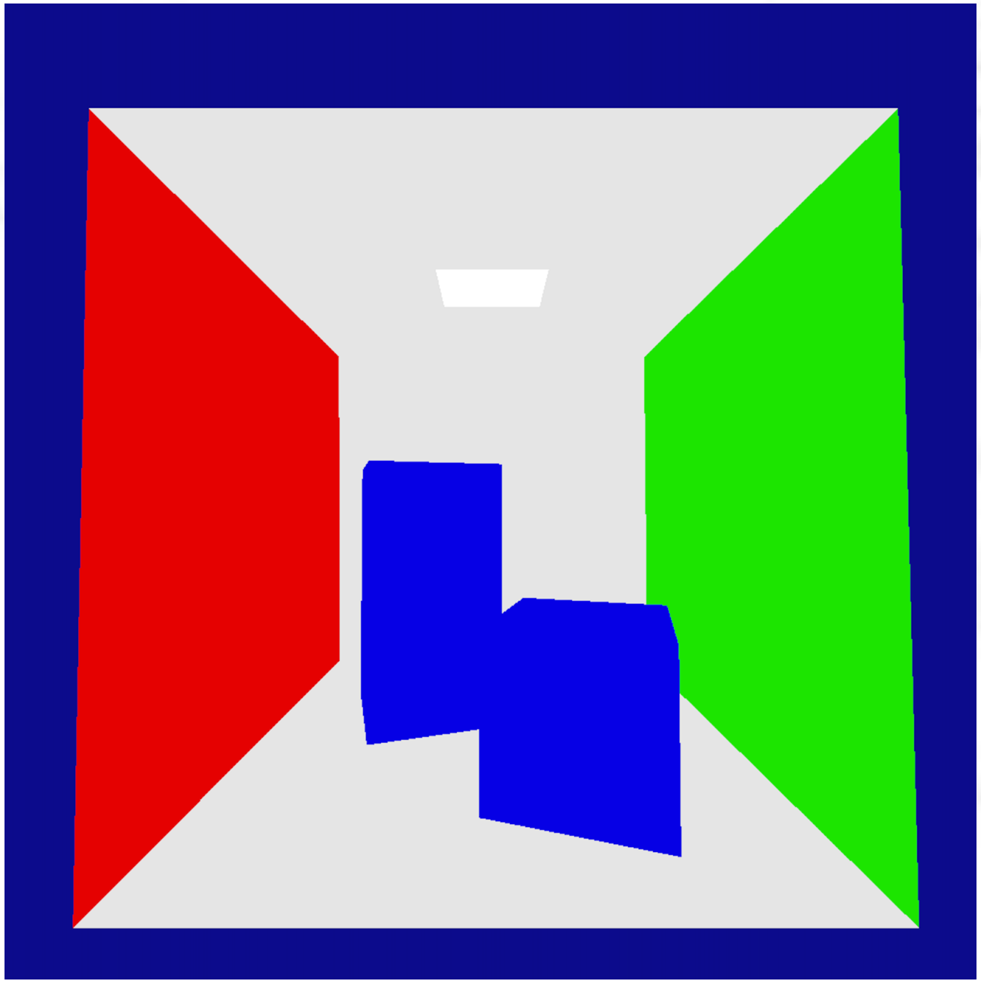
red ~(560,0,0) → (560,550,560)

light ~(213,548,227) → (343,548,332)

Code – perspective.[cpp,hpp]

- The `Perspective` class will implement a perspective camera, according to what has just been defined
- Initial files for this class are provided (`github`)

Parameterization for the Cornell Box



W = H = 1024

$\text{fovW} = 90$

fovH = fovW * H/W

$\text{eye} = (280,275,-330)$

$\text{at} = (280,265,0)$

Code – `Image.hpp` , `ImagePPM.[cpp,hpp]`

- It is proposed that on an initial approach images are saved as `.ppm` files (this is one of the simplest bitmap image formats.
- Details and code on `.ppm` files can be found at: https://www.scratchapixel.com/lessons/digital-imaging/simple-image-manipulations/reading-writing-images.html
- Initial files for these classes are provided (`github`)

[see next slide for further details]

Images and Tone Mapping

- Our renderer produces floating point values for each channel (R, G and B) of each pixel. These are positive real numbers
- The `ppm` file format only supports unsigned char values for each channel, in the set {0, 1, 2, … , 255}
- The operation of compressing the large values on an image to much smaller values, such that they can be displayed, is referred to as **Tone Mapping**
- `ImagePPM.cpp` includes the simplest (and less effective) tone mapper, such that your images can be saved. Everything should be OK if your lights accumulated power does not exceed 1.0 per channel.
- More on this later on the semester