Mestrado em
Engenharia Informática

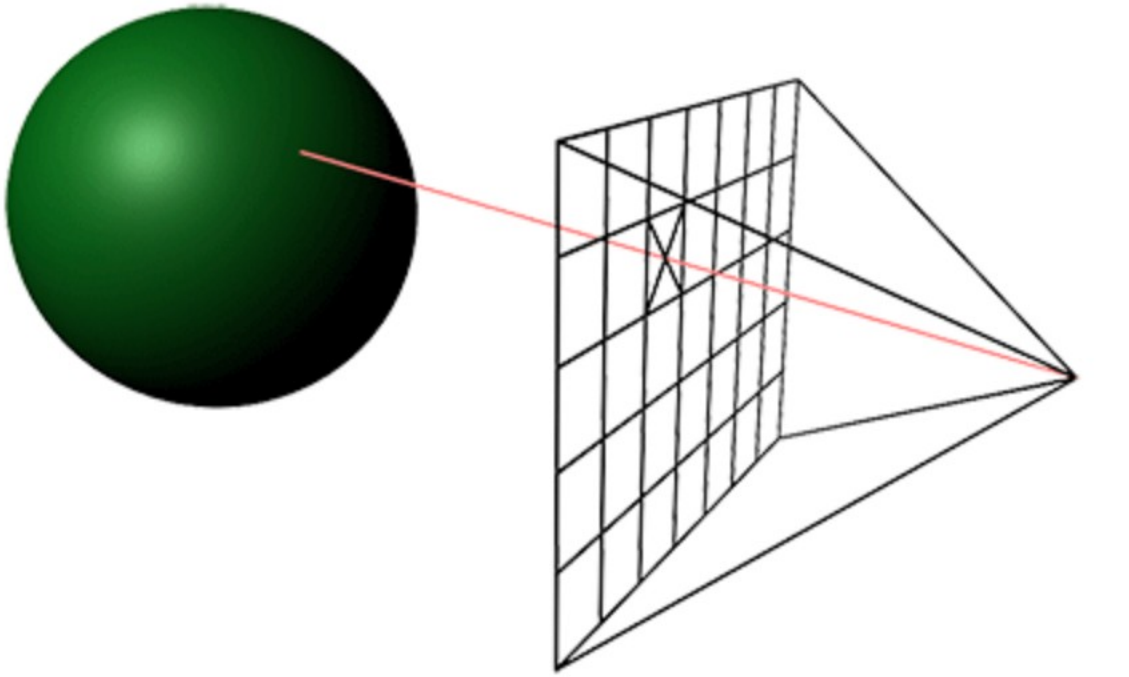
VI-RT

Perspective Camera

Image

Visualização e Iluminação

PERSPECTIVE CAMERA



Raster -> Screen -> Camera Space

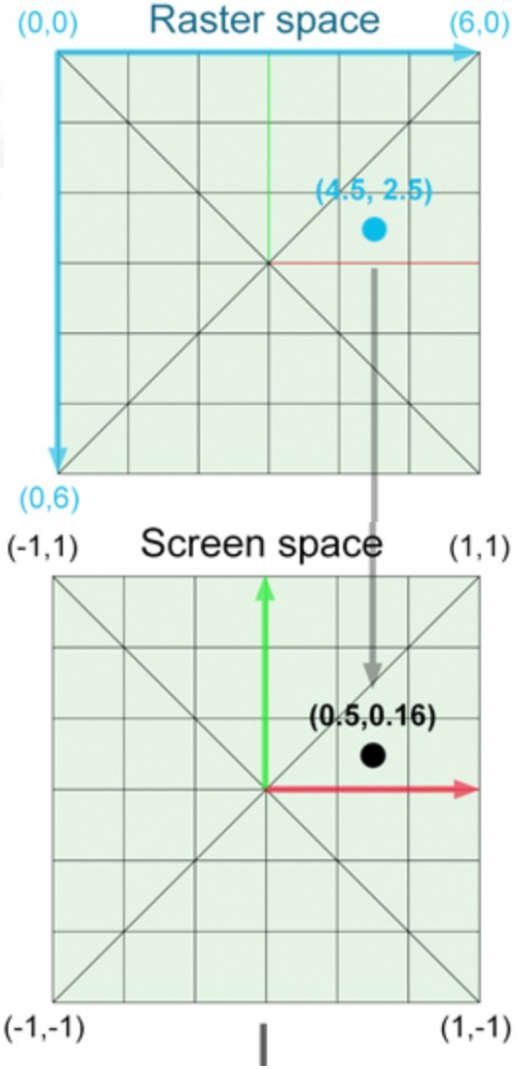


Image Resolution = (W,H)

Raster -> Screen -> Camera Space

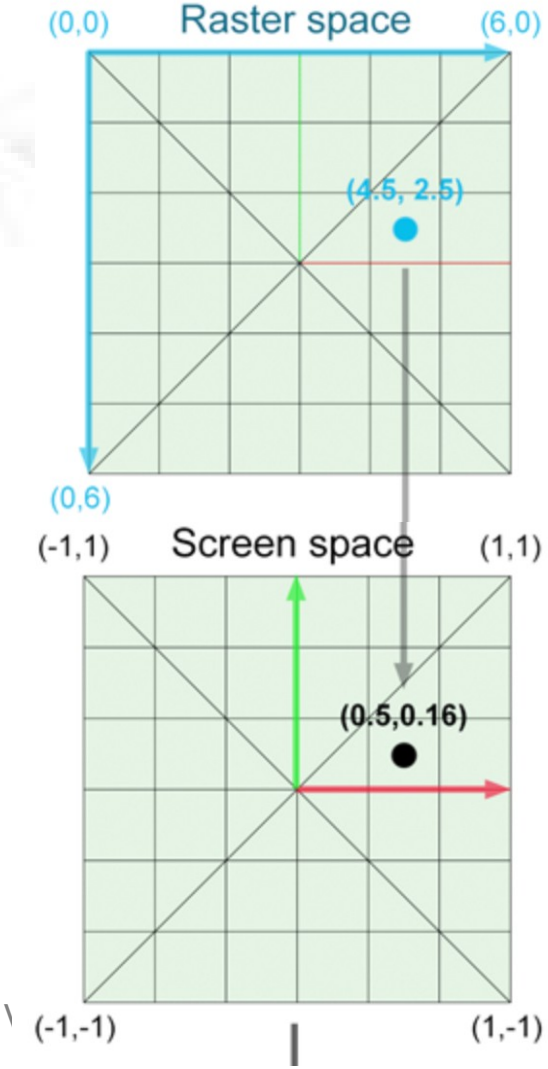


Image Resolution = (W,H)

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

Raster -> Screen -> Camera Space

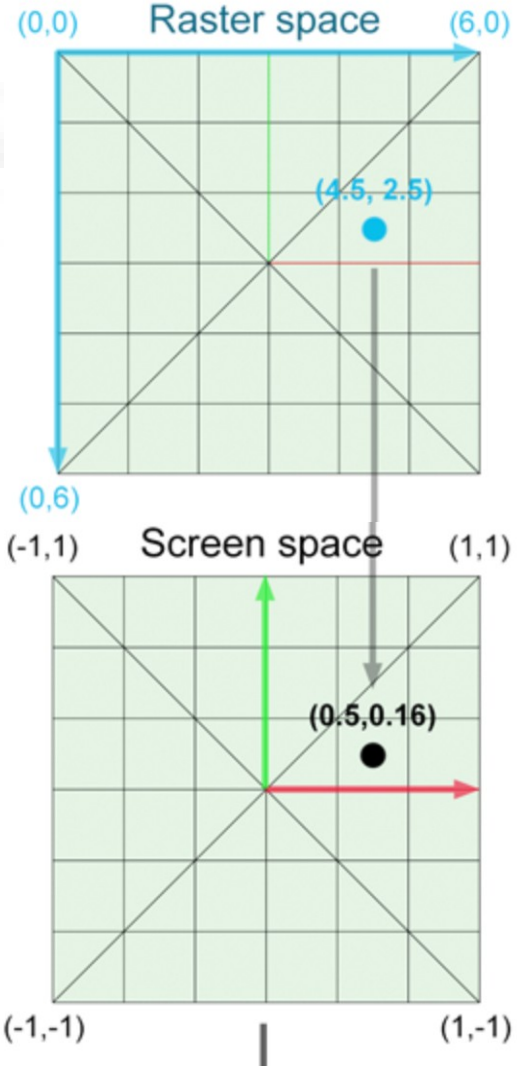
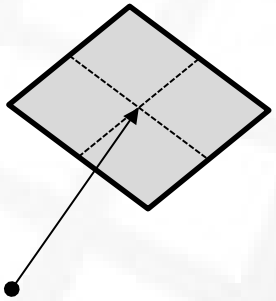


Image Resolution = (W,H)

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



Raster -> Screen -> Camera Space

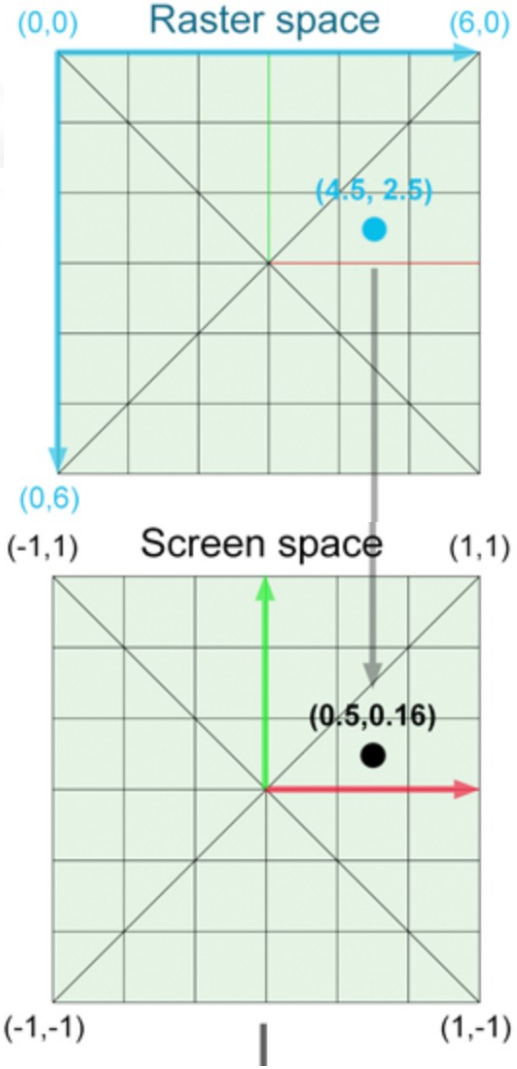
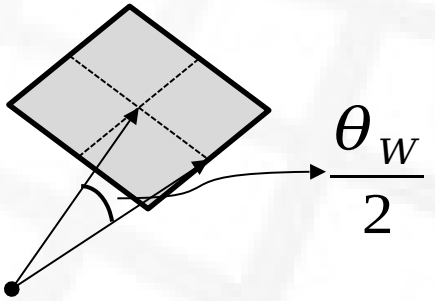


Image Resolution = (W,H)

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



Raster -> Screen -> Camera Space

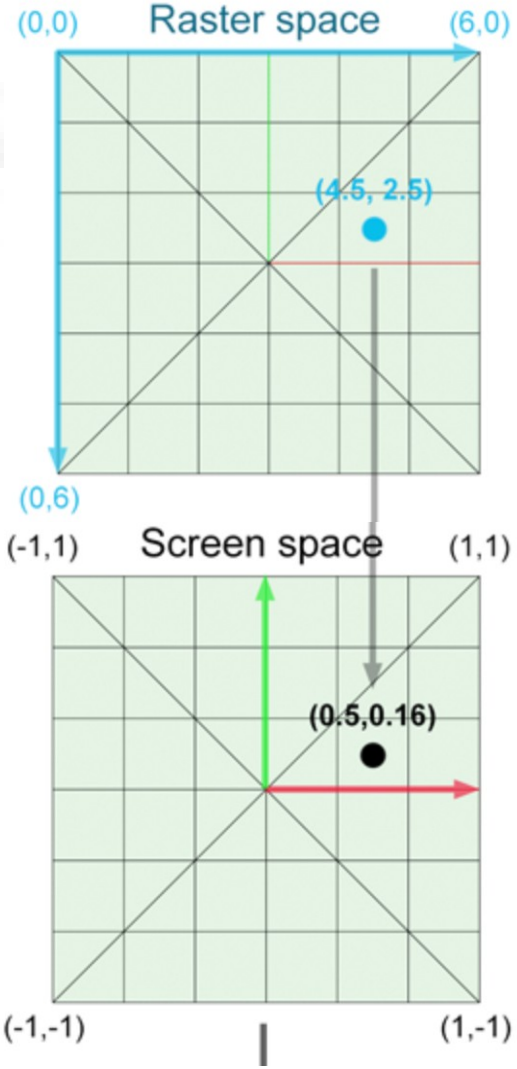
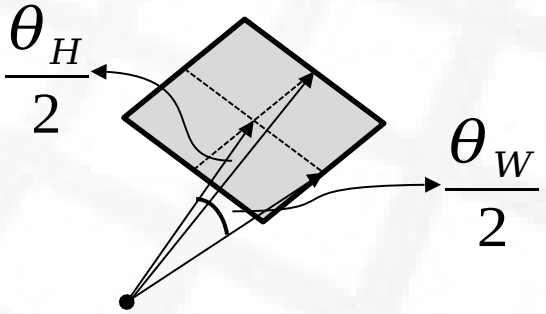


Image Resolution = (W,H)

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



Raster -> Screen -> Camera Space

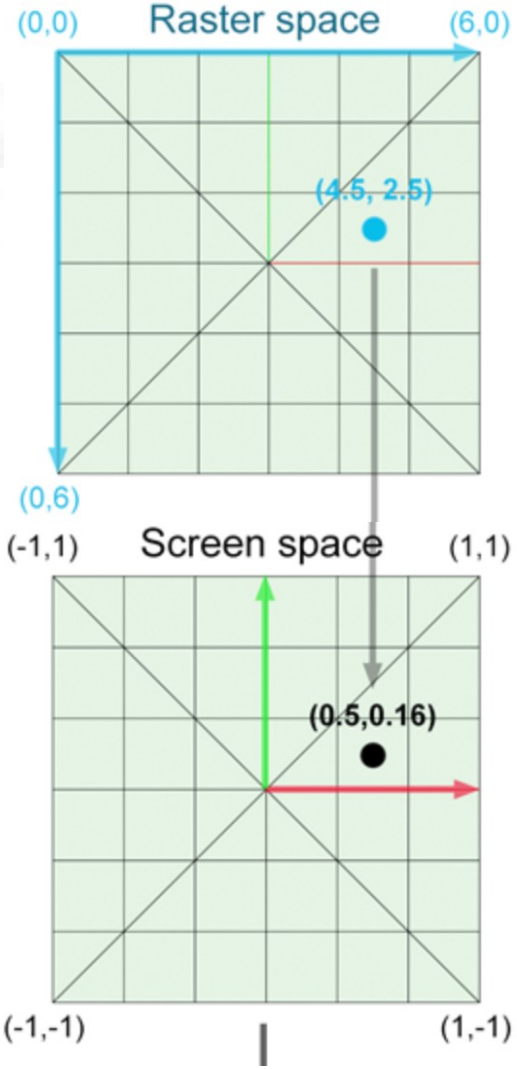
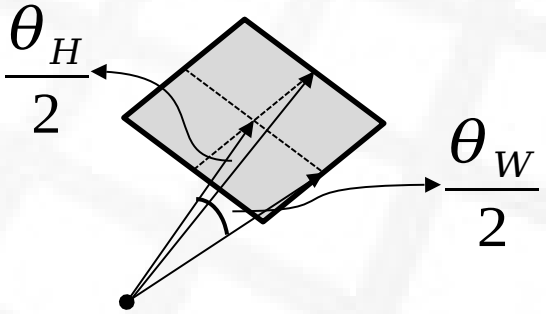


Image Resolution = (W,H)

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



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

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

Raster -> Screen -> Camera Space

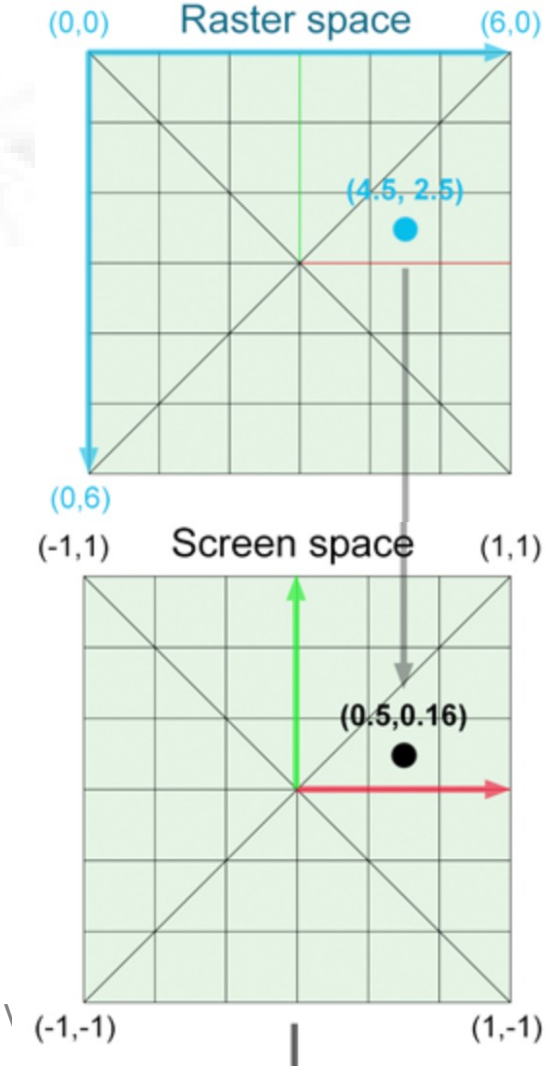
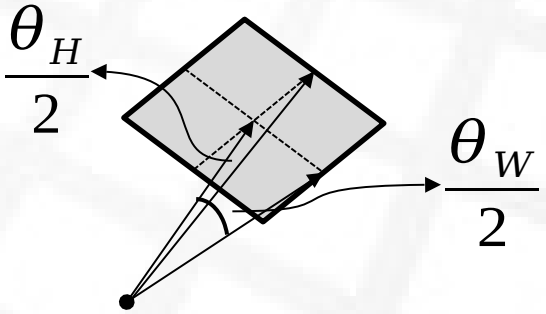


Image Resolution = (W,H)

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

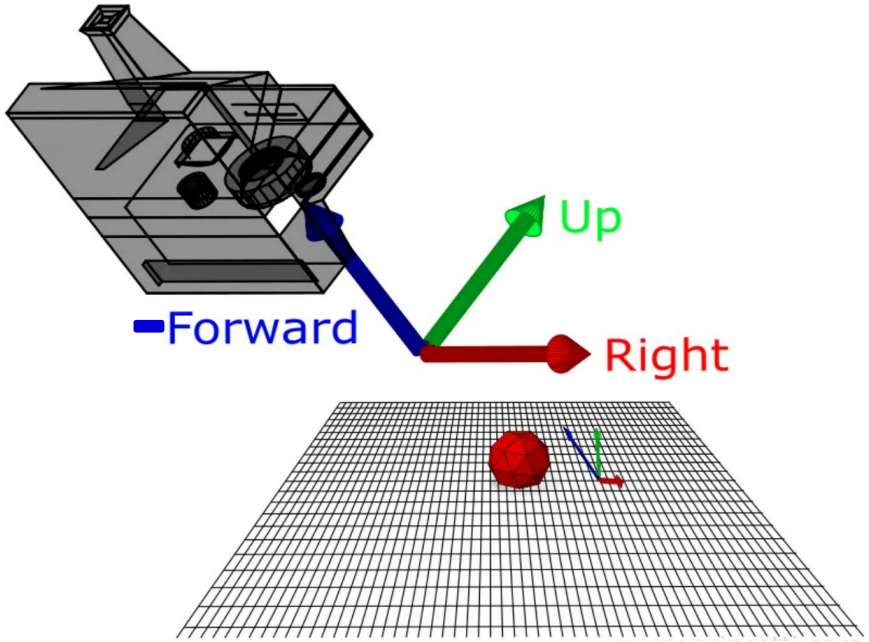


$$x_c = x_s * \tan\frac{\theta_W}{2}$$

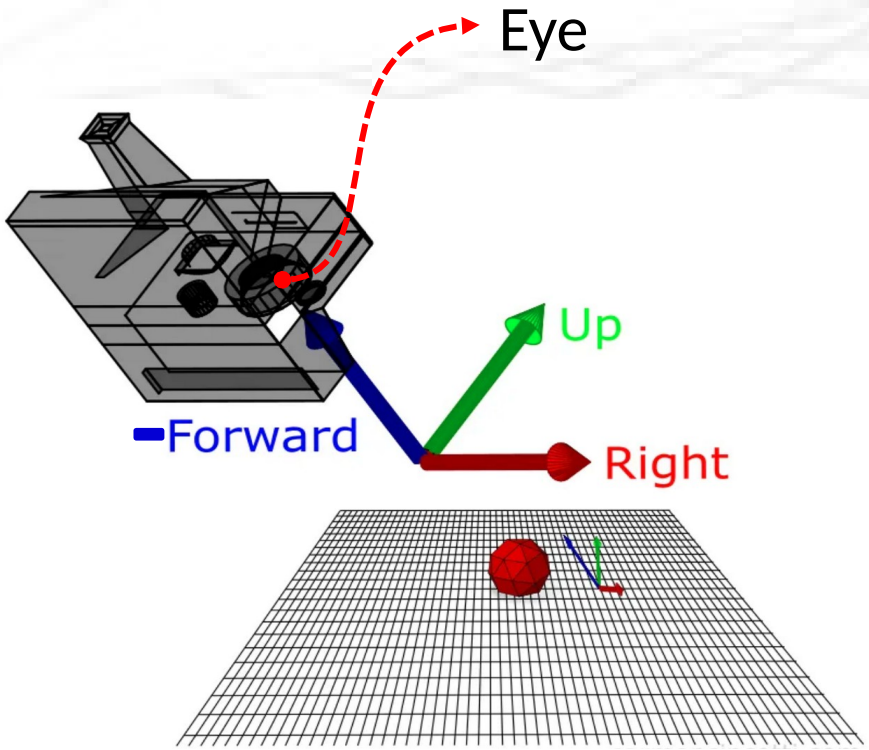
$$y_c = y_s * \tan\frac{\theta_H}{2}$$

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

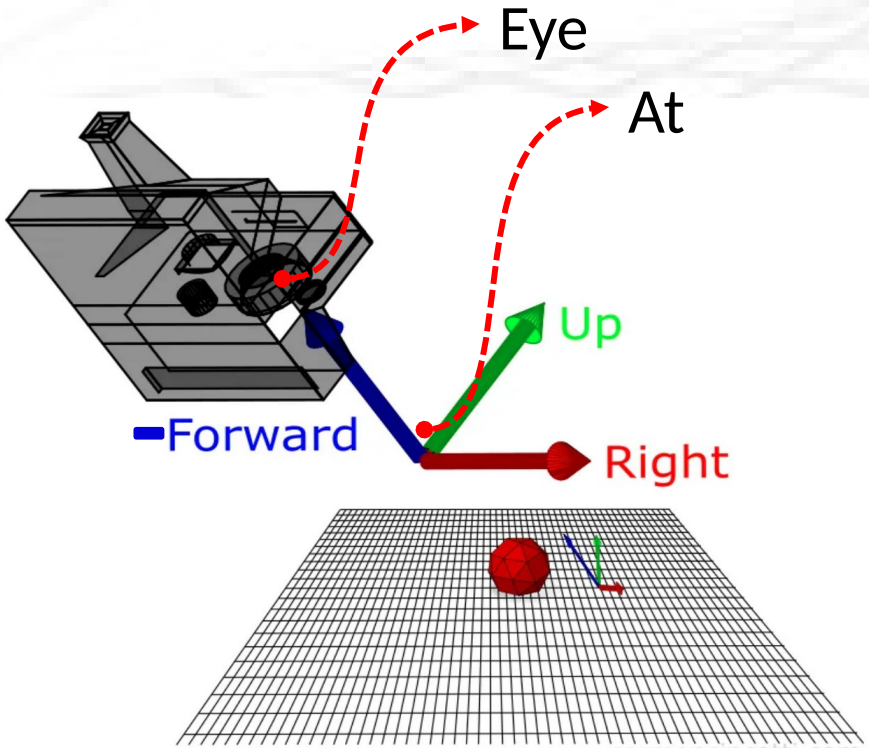
Camera Setup



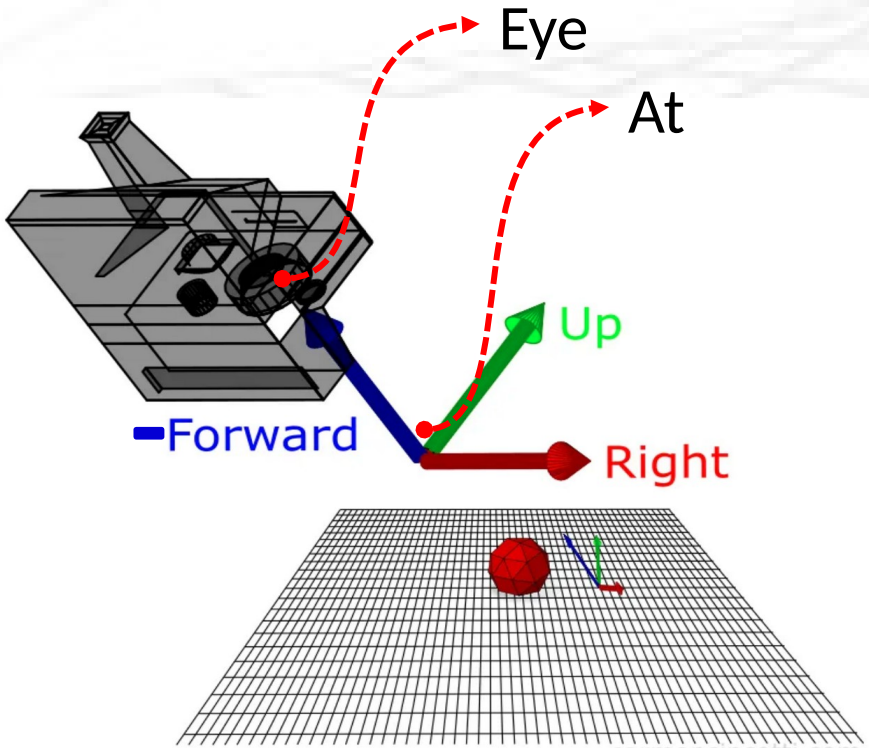
Camera Setup



Camera Setup

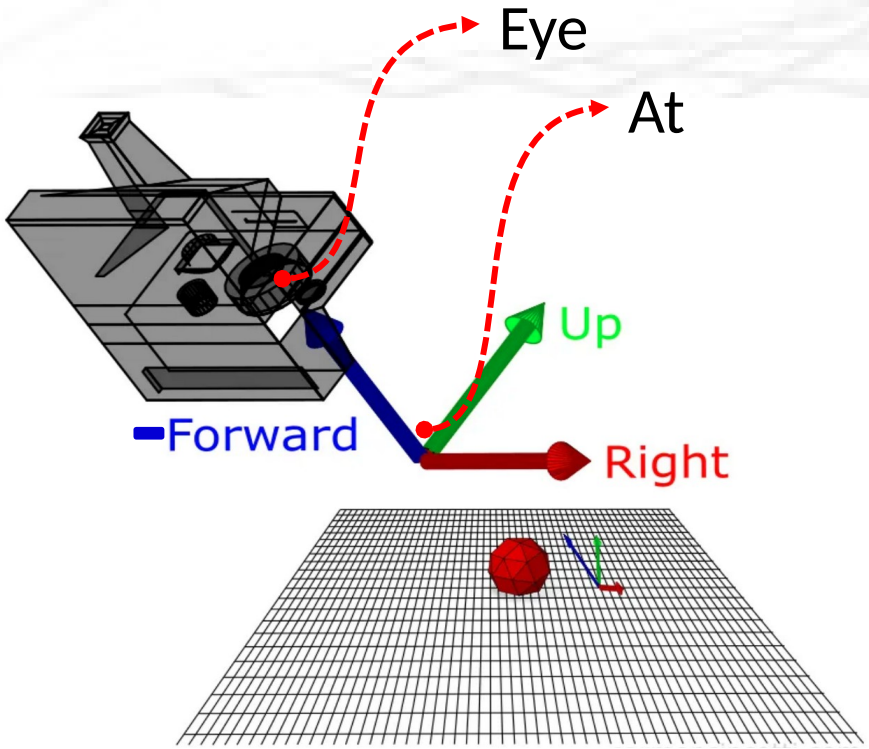


Camera Setup



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

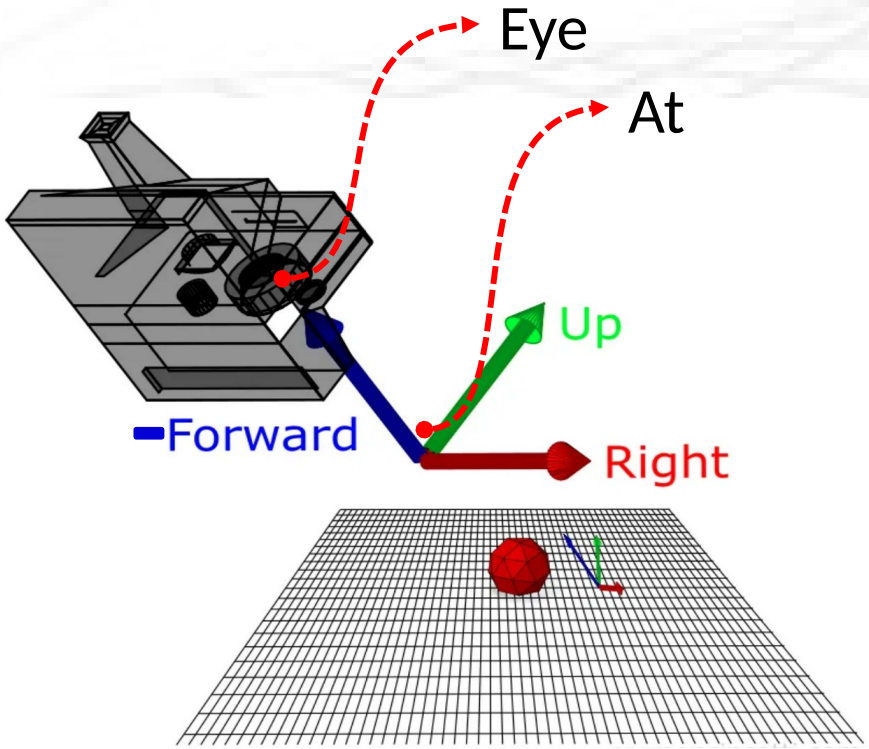
Camera Setup



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

R

Camera Setup

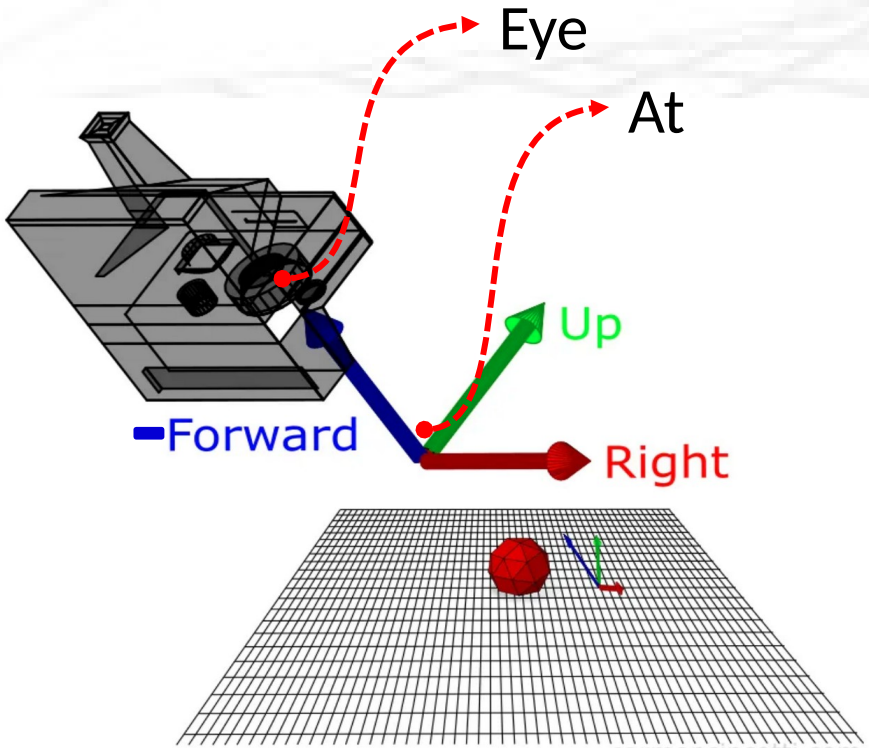


$$F = normalize(At - Eye)$$

R

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

Camera Setup



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

R

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

$$ray.o = eye$$

Code – `camera.hpp`

- This file has been changed, such that it no longer refers to classes deriving from `Image`
- Images are now created in the `main` program, passed to the `Renderer` and than eventually saved by the `main` program
- Please update this file either from `github` or from the elearning system

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 (elearning or `github`)

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 (elearning or `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

Code – main.cpp , Renderer.hpp

- The main routine was edited to reflect the fact that Image is now created and accessed outside Camera

```
int main(int argc, const char * argv[]) {

        //Load scene
        //Create ImagePPM
        //Create Perspective (camera)
        Renderer myRender (cam, &scene, img);
        myRender.Render();
        // save the image
        img->Save("MyImage.ppm");
        return 0;
}
```

- Initial files are provided (elearning or github)