Computer Graphics

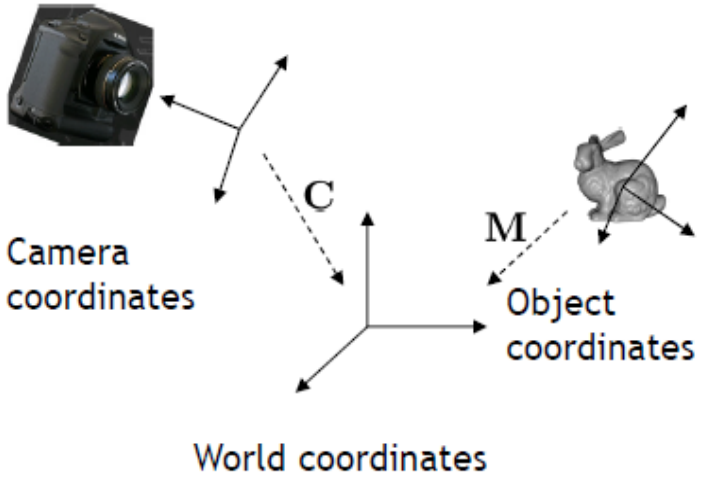
Camera

Andreas Aristidou
andarist@ucy.ac.cy
http://www.andreasaristidou.com

The scene

- A scene usually consists of a group of objects
- Each object shall have:
  - *geometric properties => the shape*
  - *material properties = > how light reflects, how and if it emits light*

- In computer graphics, we usually use three coordinate systems:
  - World coordinate system.
  - Camera coordinate system.
  - Object coordinate system

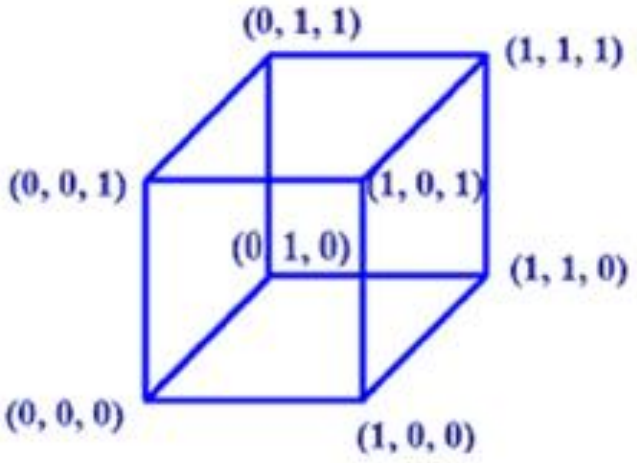


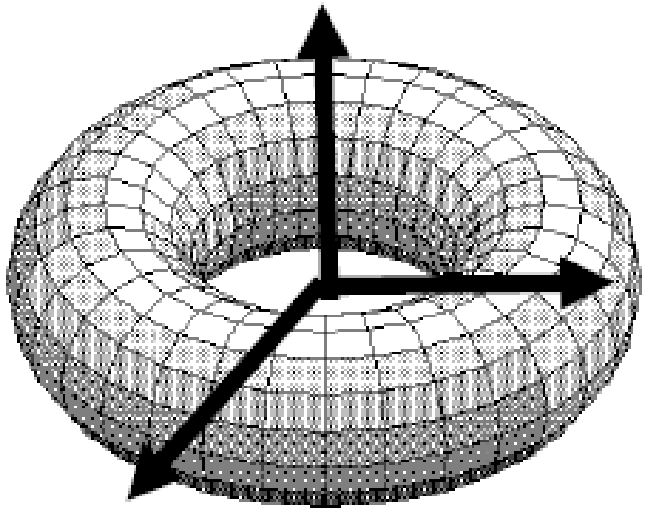
World coordinate system

- Common reference for all objects on stage
- There is no template for the orientation of the coordinate system
  - If there is a plane, usually x/y is horizontal and z points upwards (height)
  - In graphics (e.g. OpenGL, Maya, Unity, Blender) x/y is the level of the screen, and z shows (is perpendicular) to the screen

Object coordinate system

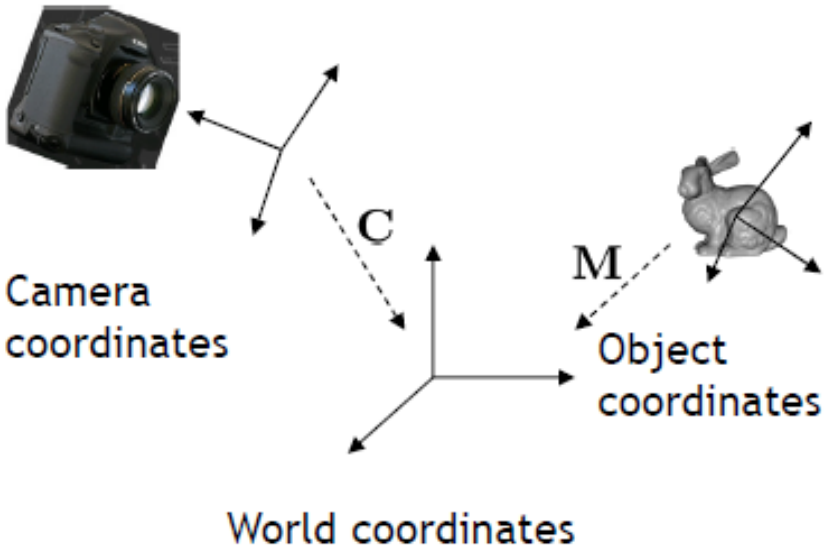
- Local coordinates at which the points (e.g., the geometry) of objects are given.
- The origin of the axes, often, is in the geometric center, at the base or at a corner of the object.
- It depends on how the object is created or used.
- The choice can be determined based on the **convenience of the creator.**





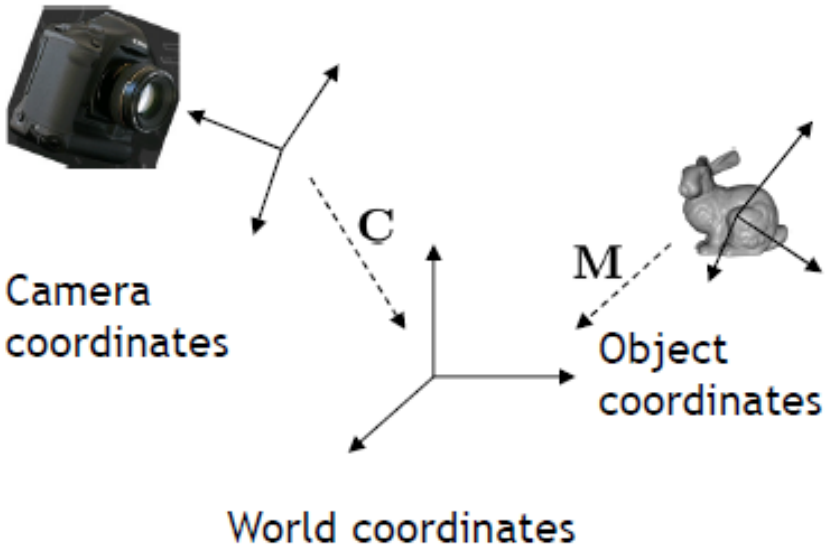
Transformation of an object

- The transformation from an object to the coordinates of our virtual world is different for each object.
  - Specifies the placement of the object in a scene.
  - Given by the "Model-to-world transformation" **M** (see the scene graph).



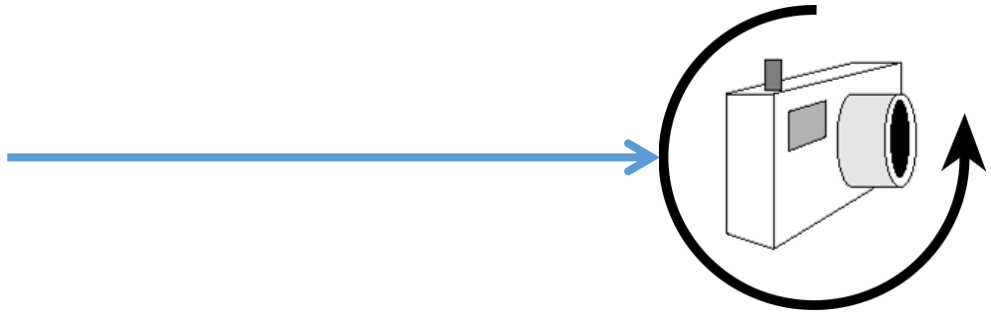
Camera coordinate system

- The origin of the axes, defines the **center of projection** of the camera.
- The plane $x - y$ is parallel to the plane of the image.
- The axis z is perpendicular to the image plane.
- The camera-to-world transformation matrix, **C** specifies the transfer from the coordinates of the camera to the world coordinates.
  - Placing the camera in the world



Specify the camera coordinates for the projection

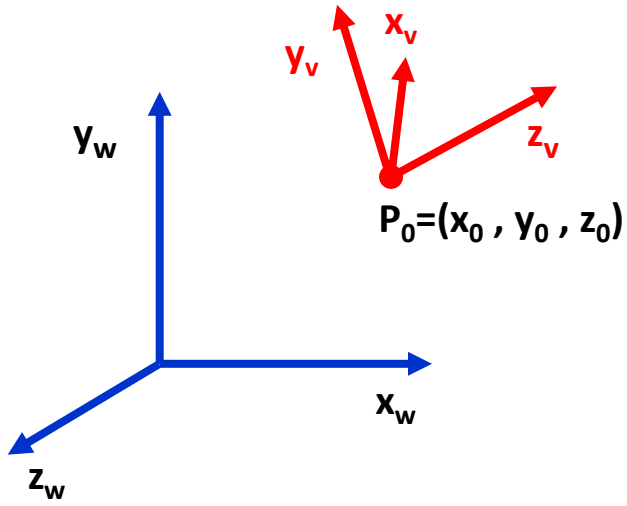
- A scene has a camera/projection point from which the scene is projected
- The camera has a **location** and an **orientation** in the 3D space.

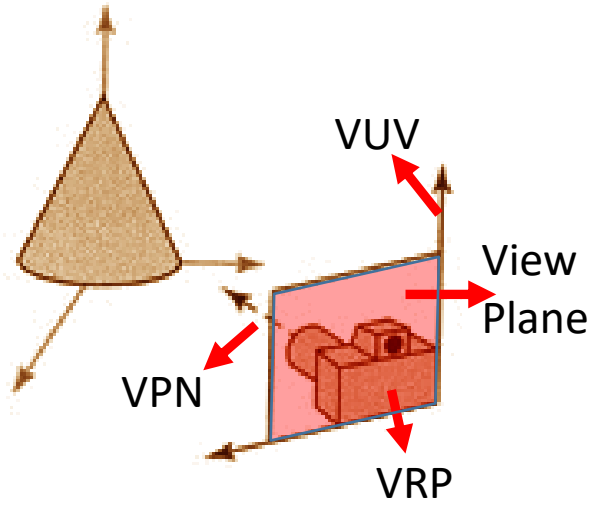


- How can we express the coordinates of the object in relation to the coordinates of the world or vice versa?
  - We should know the position of the camera in the world coordinate system.
  - We can calculate the camera-to-world transformation matrix, using the orientation and translation of the camera from the origin of the axes.

Specify the camera coordinates for the projection

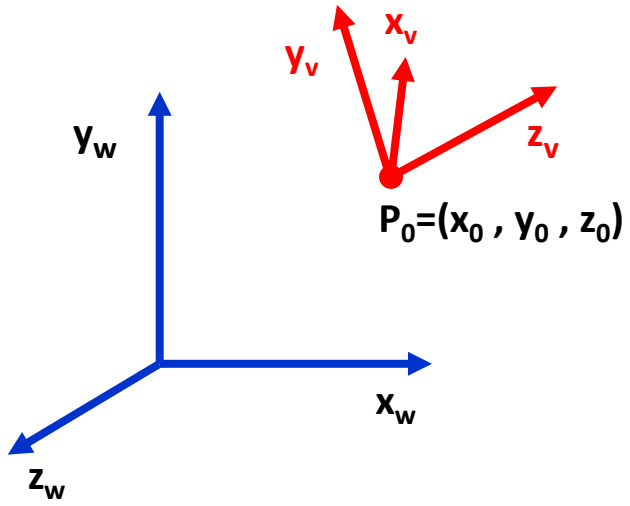
- **View Reference Point, VRP** or **P_0**
  - Reference point by which you set the position of the camera
  - Center of Projection (COP), the location of the “eye” of the camera.
- **View Plane Normal, VPN** or **N**
  - Where camera is looking at.
- **View Up Vector, VUV** or **V**
  - The angle of the camera around the VPN

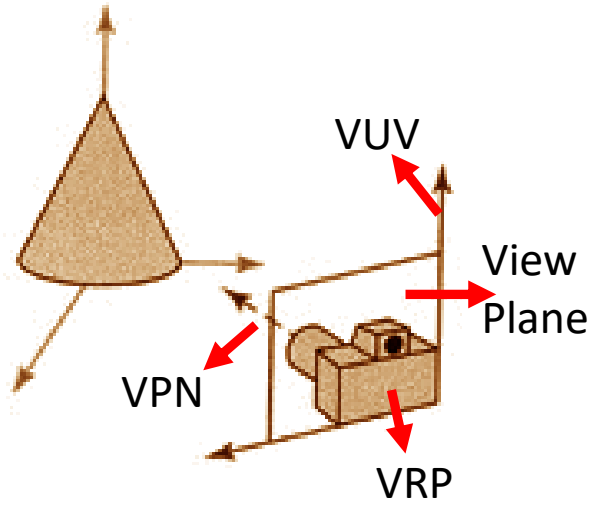




Specify the camera coordinates for the projection

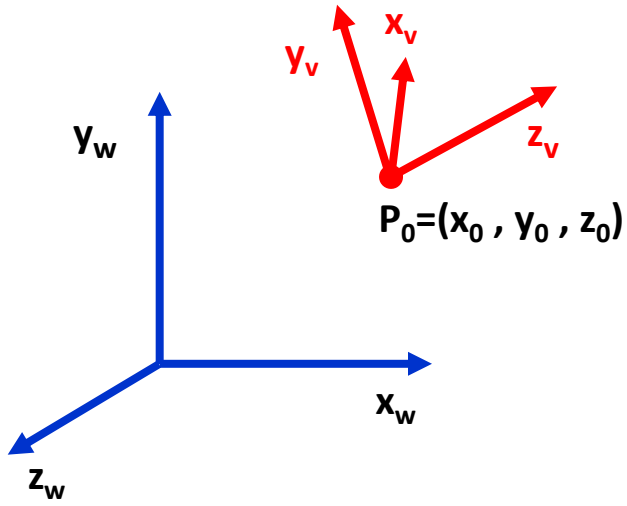
- For a specific image of a scene we define the **viewing-coordinate system**.
- A **view-plane** (or **projection plane**) is defined perpendicular to the z-axis view.
- The global coordinates are converted into projection coordinates, and then the projection coordinates are projected on the projection plane (view plane).

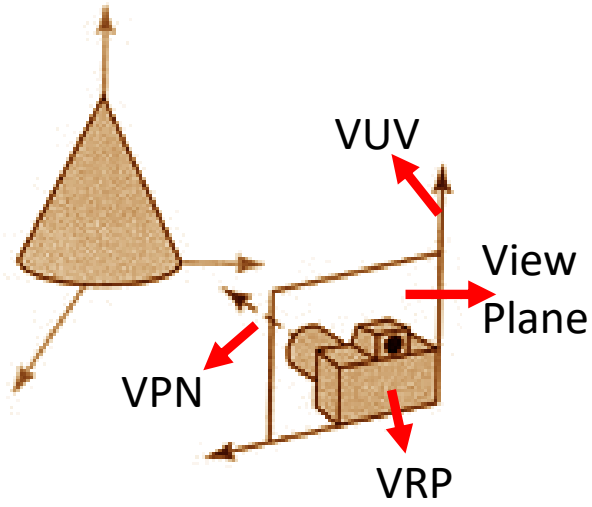




Specify the camera coordinates for the projection

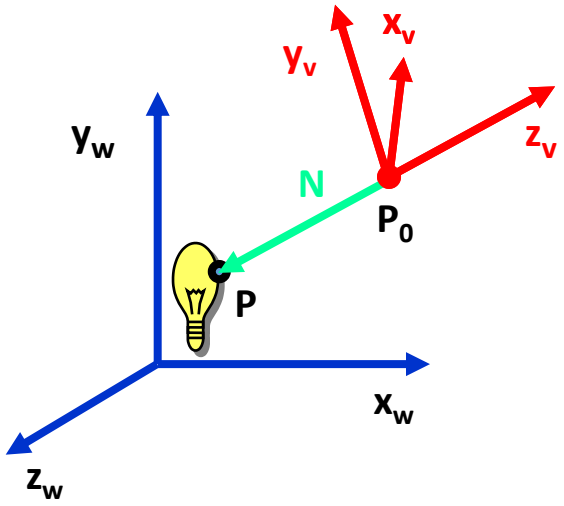
- To create a **view-plane**, first we choose the **view reference point, P_0** where we want to project.
- This point is the origin of our coordinate system.





Specify the camera coordinates for the projection

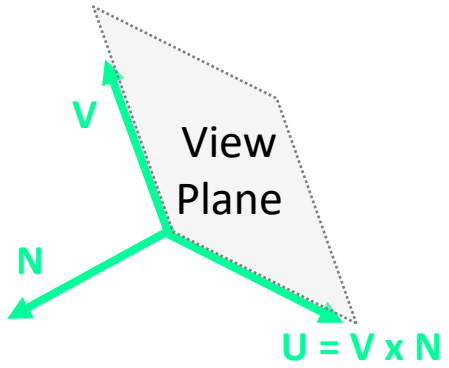
- Then, we want to choose the projection of axis z (at the same time choosing the orientation of the view-plane), by defining the **view-plane normal vector, N**.
- Consequently, we choose a global coordinate position **P** and this point determines the direction of the **N**.
  - *OpenGL* determines the direction of **N** using the point **P** as the point that shows the relation with the source of view coordinates.

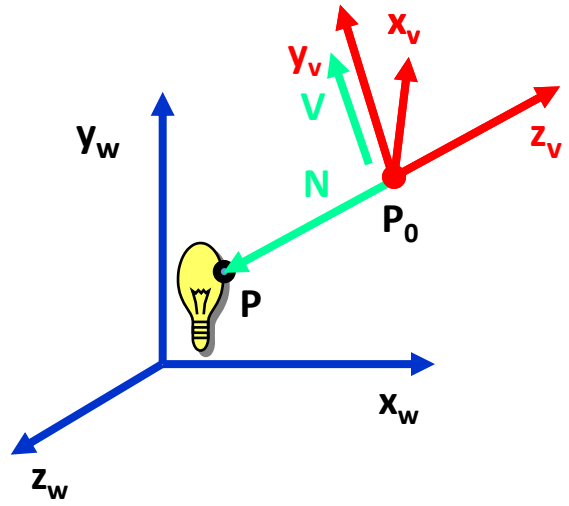


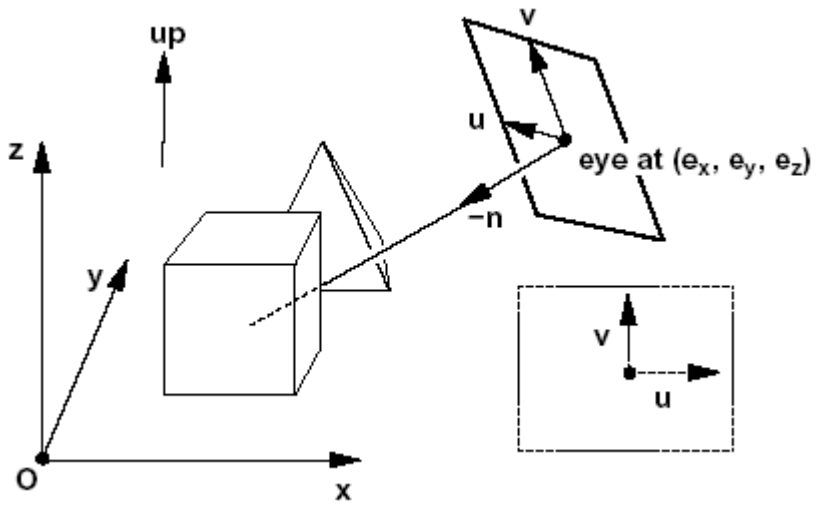
If we choose a point in an object, we can think of that point as the position where we are targeting the camera to take a photo of the object.

Specify the camera coordinates for the projection

- Then, we choose the viewing direction of the projection, by defining the vector **view-up, V**.
  - this vector is used to determine the positive direction for the axis y_v.
- The vector **V** is perpendicular to **N**.
- Using vectors **N** and **V**, we can calculate the vector **U**, which is perpendicular to **N** and **V**, and defines the direction of the axis x_v.

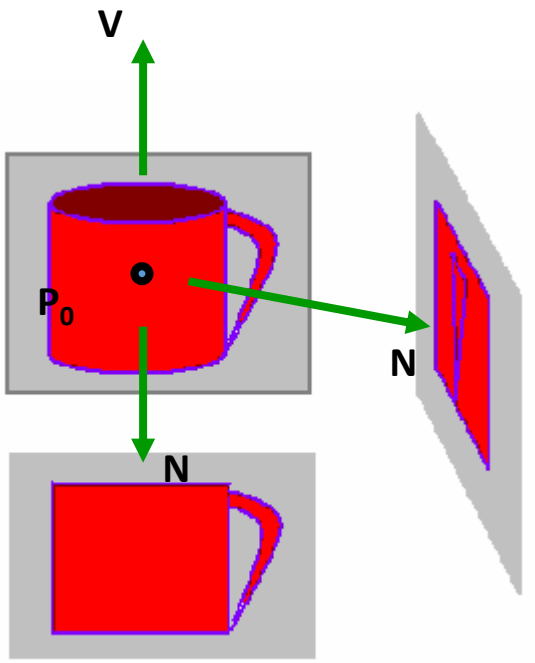






Specify the camera coordinates for the projection

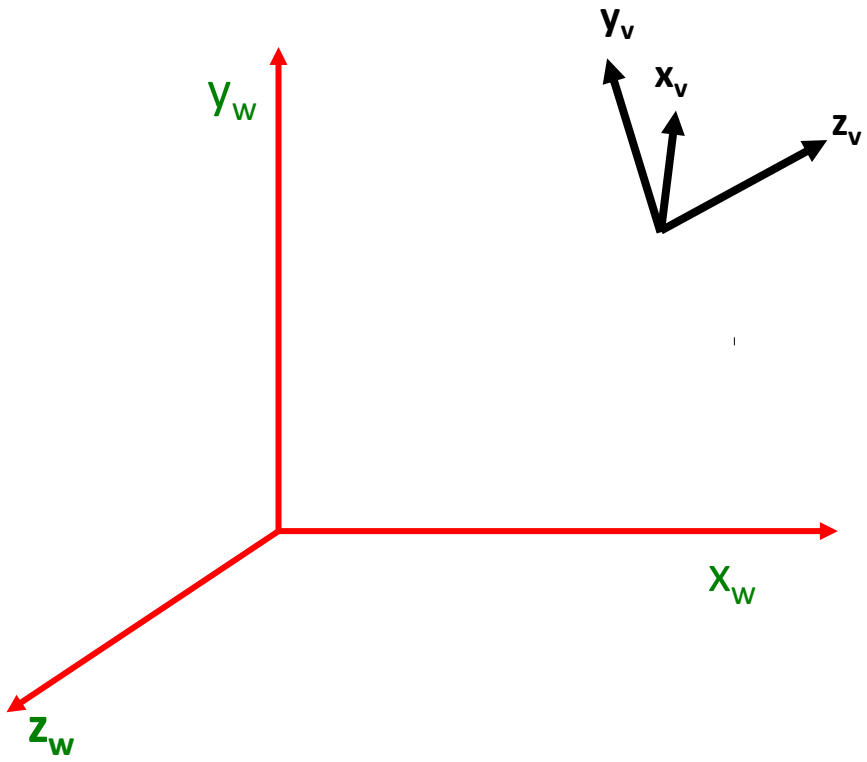
- To create a series of views of a scene, we can keep the projection point constant, and change the direction of the **N**.
- This corresponds to the creation of views as we move around the view coordinates.



World coordinate system (WC) and View Coordinates (VC)

Transforming from WC to VC

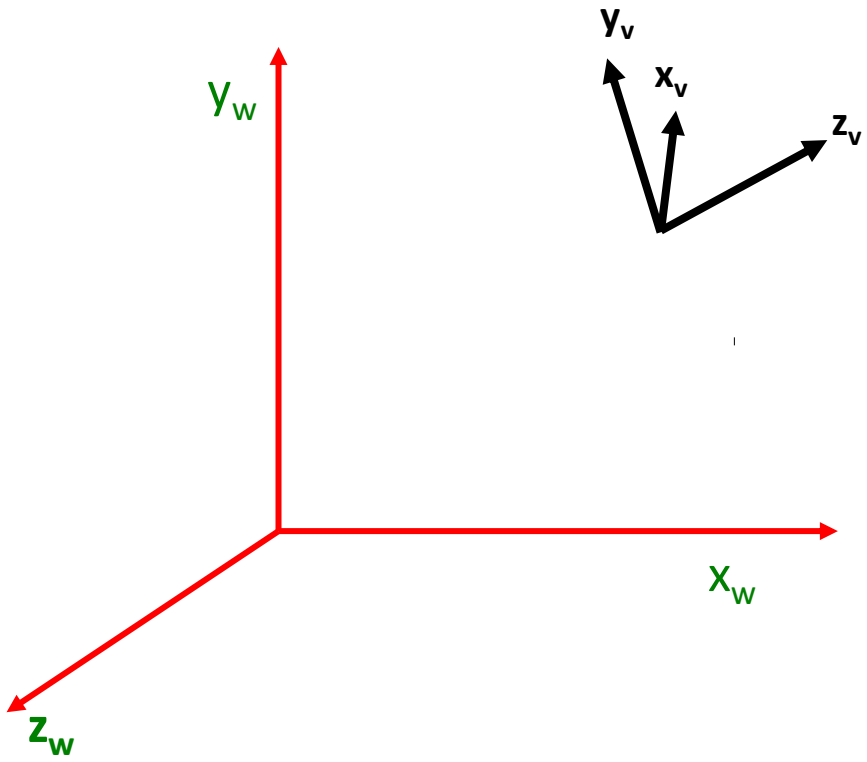
- Transforming the coordinates of the object from the world coordinate system to the viewer's coordinate system, is equivalent to a transformation involving the translation and rotation of the object.



World coordinate system (WC) and View Coordinates (VC)

Transforming from WC to VC

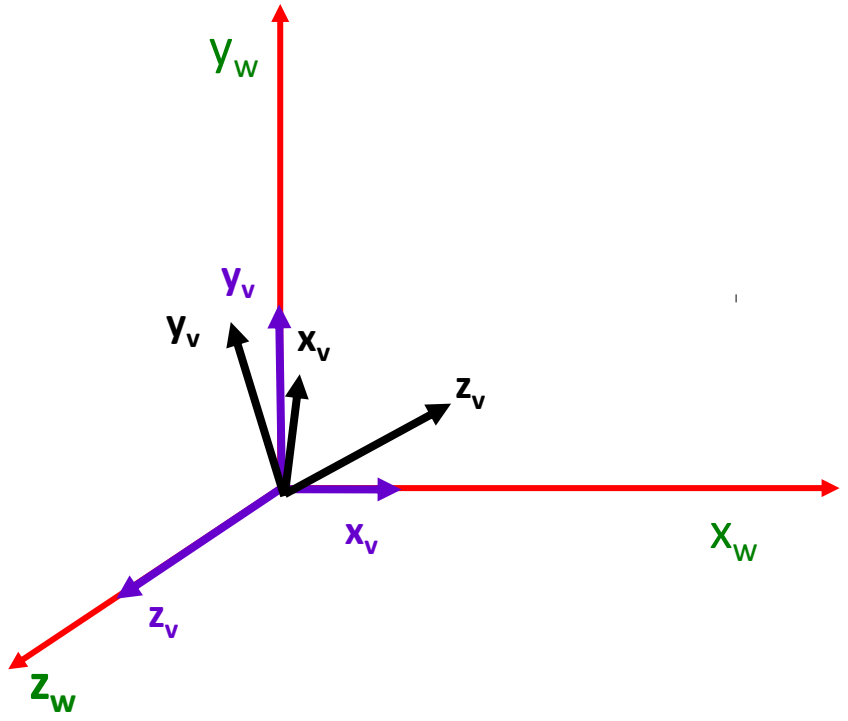
- First, we translate (move) the reference point of the view coordinate system (projection's) to the origin of the world coordinate system.



World coordinate system (WC) and View Coordinates (VC)

Transforming from WC to VC

- Then, we rotate the axes x_v, y_v and z_v , so that they are aligned with the axes x_w, y_w and z_w, respectively.



World coordinate system (WC) and View Coordinates (VC)

Transforming from WC to VC

- If the point in the VC is defined in the WC as (x_0, y_0, z_0), then that point is translated (moved) in the WC using the transformation **T.**

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & -x_0 \\ 0 & 1 & 0 & -y_0 \\ 0 & 0 & 1 & -z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

World coordinate system (WC) and View Coordinates (VC)

Transforming from WC to VC

- Accordingly, the rotation requires 3 transformations of the coordinate axis, depending on the direction **N**.
  - On the axes of the x, y, z

$$\mathbf{R}_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & Cos\theta & -Sin\theta & 0 \\ 0 & Sin\theta & Cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{R}_y = \begin{bmatrix} Cos\alpha & 0 & Sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ -Sin\alpha & 0 & Cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{R}_z = \begin{bmatrix} Cos\beta & -Sin\beta & 0 & 0 \\ Sin\beta & Cos\beta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

World coordinate system (WC) and View Coordinates (VC)

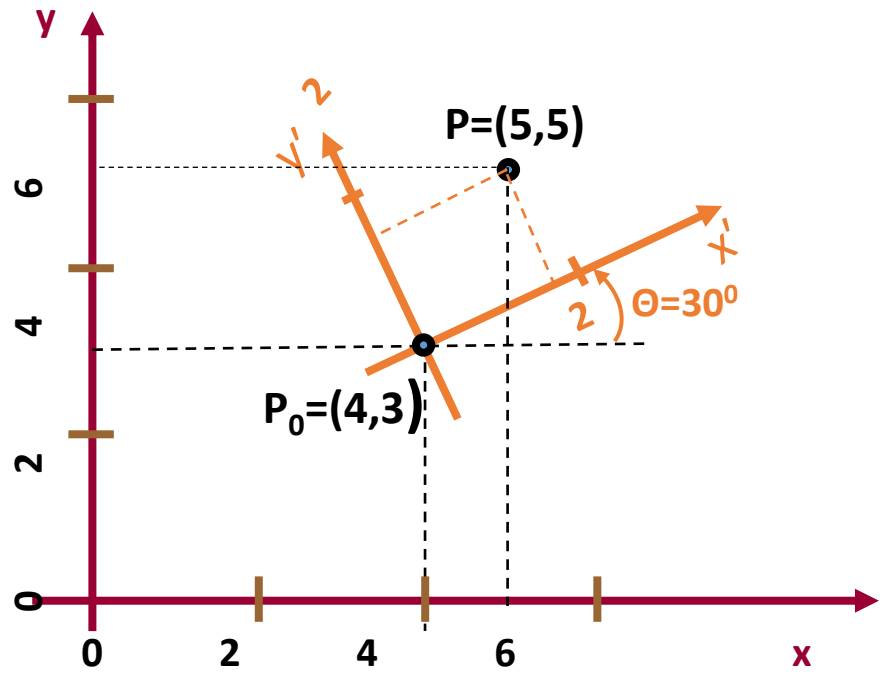
Transforming from WC to VC

- The complete transformation from the world coordinate system to the coordinate system of the observer (VC) is given by the transformation:

$$\mathbf{M}_{wc,vc} = \mathbf{R}_z \cdot \mathbf{R}_y \cdot \mathbf{R}_x \cdot \mathbf{T}$$

World coordinate system (WC) and View Coordinates (VC)

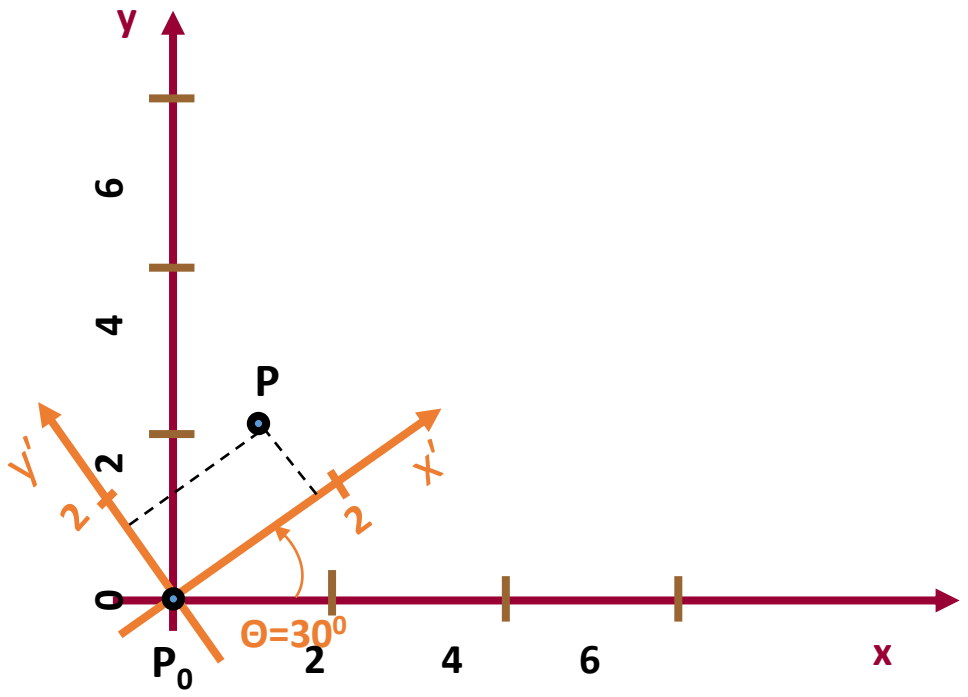
An example for 2D



World coordinate system (WC) and View Coordinates (VC)

Translation:

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & -4 \\ 0 & 1 & -3 \\ 0 & 0 & 1 \end{bmatrix}$$

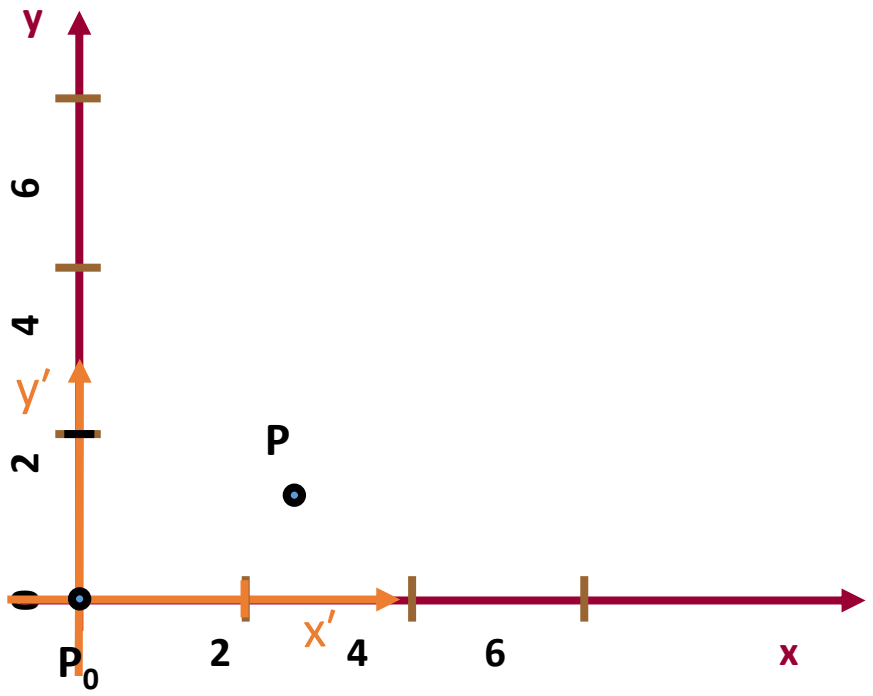


World coordinate system (WC) and View Coordinates (VC)

Rotation:

$$\mathbf{R} = \begin{bmatrix} \cos(-30) & -\sin(-30) & 0 \\ \sin(-30) & \cos(-30) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0.8660 & 0.5 & 0 \\ -0.5 & 0.8660 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



World coordinate system (WC) and View Coordinates (VC)

The new coordinates

$$\mathbf{M}_{wc.vc} = \begin{bmatrix} 0.866 & 0.5 & 0 \\ -0.5 & 0.866 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -4 \\ 0 & 1 & -3 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 0.866 & 0.500 & -4.964 \\ -0.500 & 0.866 & -0.598 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 5 \\ 5 \\ 1 \end{bmatrix} = \begin{bmatrix} 1.866 \\ 1.232 \\ 1 \end{bmatrix}$$

World coordinate system (WC) and View Coordinates (VC)

- Another method for creating the rotation transformation matrix is to calculate the vectors ***uvn***, that will give us directly the rotation matrix. Having the vectors **N** and **V**, then we can calculate the ***n, u, v*** as follows

$$\mathbf{n} = \frac{\mathbf{N}}{|\mathbf{N}|} = (n_1, n_2, n_3)$$

$$\mathbf{u} = \frac{\mathbf{V} \times \mathbf{N}}{|\mathbf{V} \times \mathbf{N}|} = (u_1, u_2, u_3)$$

$$\mathbf{v} = \mathbf{n} \times \mathbf{u} = (v_1, v_2, v_3)$$

World coordinate system (WC) and View Coordinates (VC)

- this method also automatically adjusts the direction (orientation) of the **V** so that the **v** is perpendicular to the **n**. The rotation matrix is:

$$\mathbf{R} = \begin{bmatrix} u_1 & u_2 & u_3 & 0 \\ v_1 & v_2 & v_3 & 0 \\ n_1 & n_2 & n_3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

World coordinate system (WC) and View Coordinates (VC)

- As a reminder, the translation is given by the transformation **T.**

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & -x_0 \\ 0 & 1 & 0 & -y_0 \\ 0 & 0 & 1 & -z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

World coordinate system (WC) and View Coordinates (VC)

- So the overall transformation is given by the matrix:

$$\mathbf{M}_{WC,VC} = \begin{bmatrix} u_1 & u_2 & u_3 & t_1 \\ v_1 & v_2 & v_3 & t_2 \\ n_1 & n_2 & n_3 & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

όπου

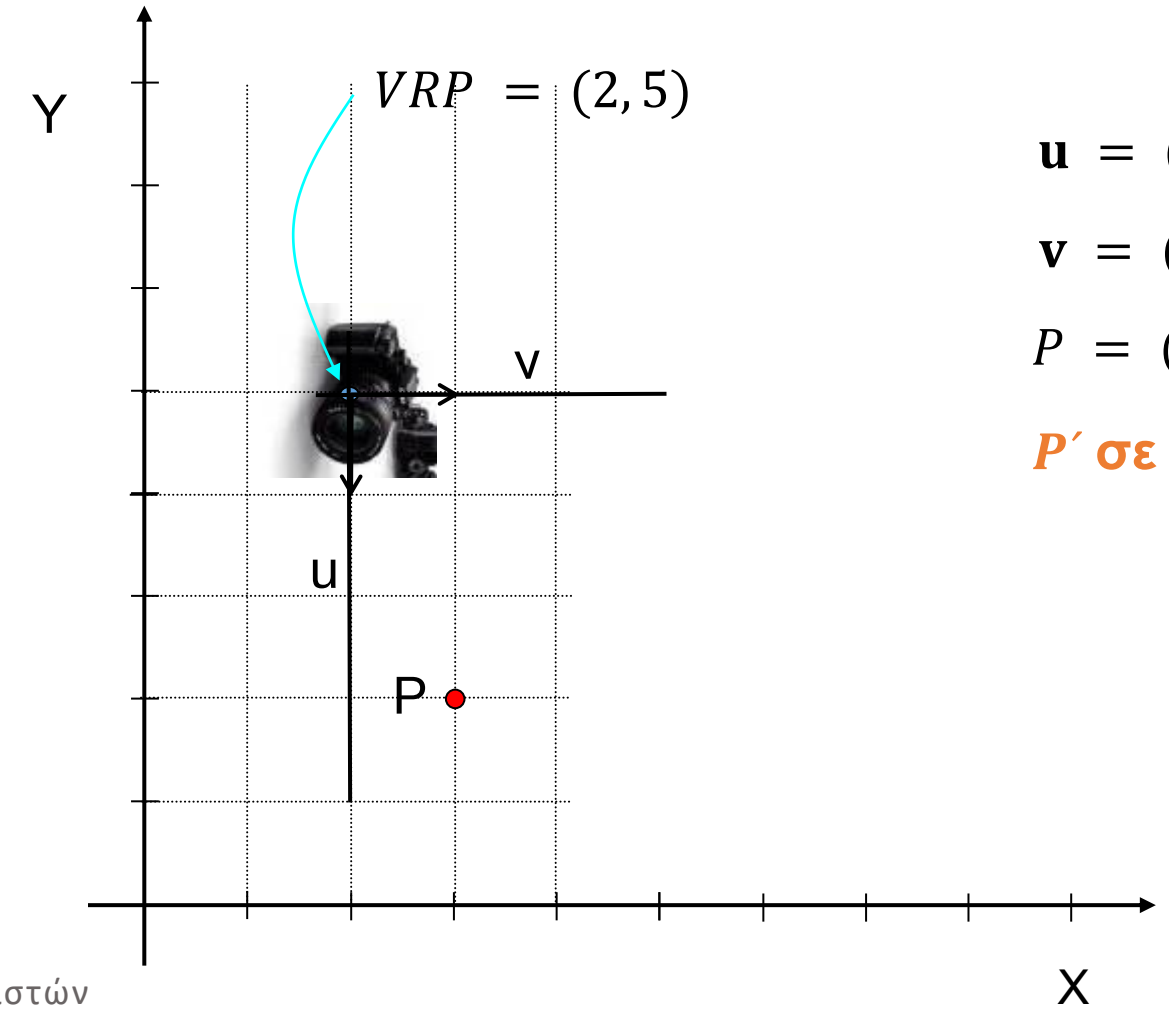
$$t_1 = -\mathbf{u} \cdot \mathbf{P}_0 = -x_0 u_x - y_0 u_y - z_0 u_z$$

$$t_2 = -\mathbf{v} \cdot \mathbf{P}_0 = -x_0 v_x - y_0 v_y - z_0 v_z$$

$$t_3 = -\mathbf{n} \cdot \mathbf{P}_0 = -x_0 n_x - y_0 n_y - z_0 n_z$$

World coordinate system (WC) and View Coordinates (VC)

An example for 2D



$\mathbf{u} = (0,-1)$

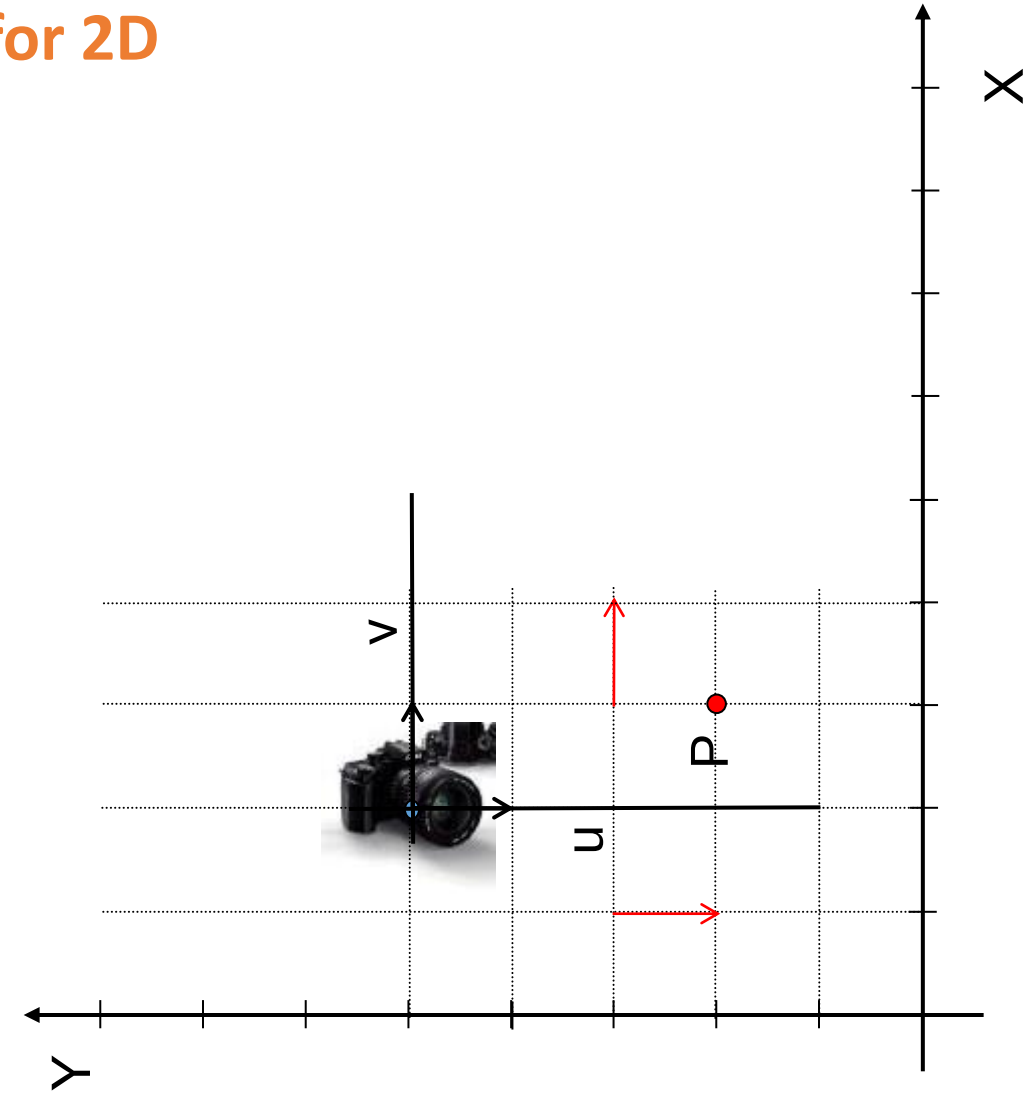
$\mathbf{v} = (1,0)$

$P = (3,2)$ σε WC

***P΄* σε VC = ??**

World coordinate system (WC) and View Coordinates (VC)

An example for 2D



$VRP = (2, 5)$

$\mathbf{u} = (0, -1)$

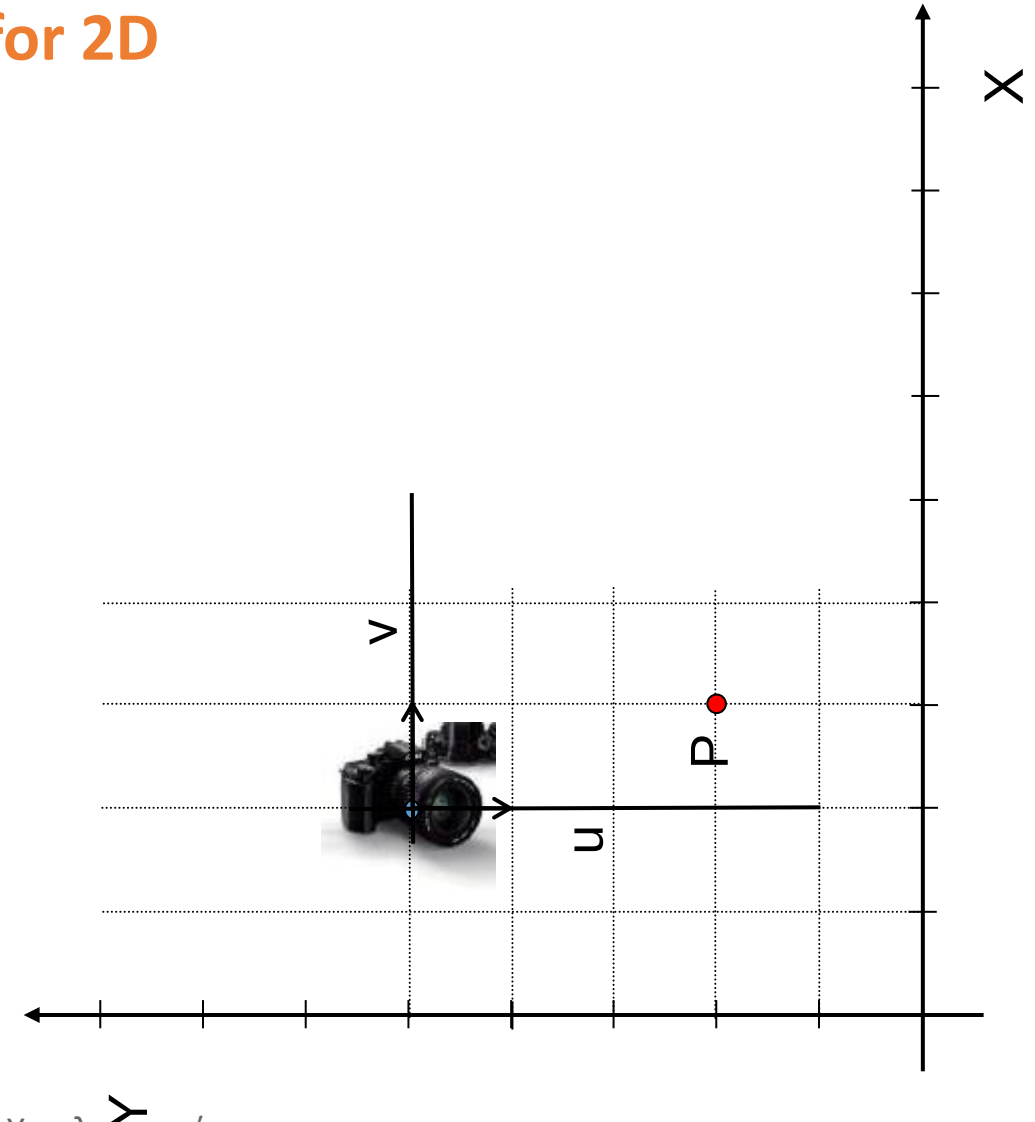
$\mathbf{v} = (1, 0)$

$P = (3,2)$ σε WC

***P´* to VC = ??**

World coordinate system (WC) and View Coordinates (VC)

An example for 2D



$$VRP = (2,5)$$

$$\mathbf{u} = (0,-1)$$

$$\mathbf{v} = (1,0)$$

$$P = (3,2) \text{ σε WC}$$

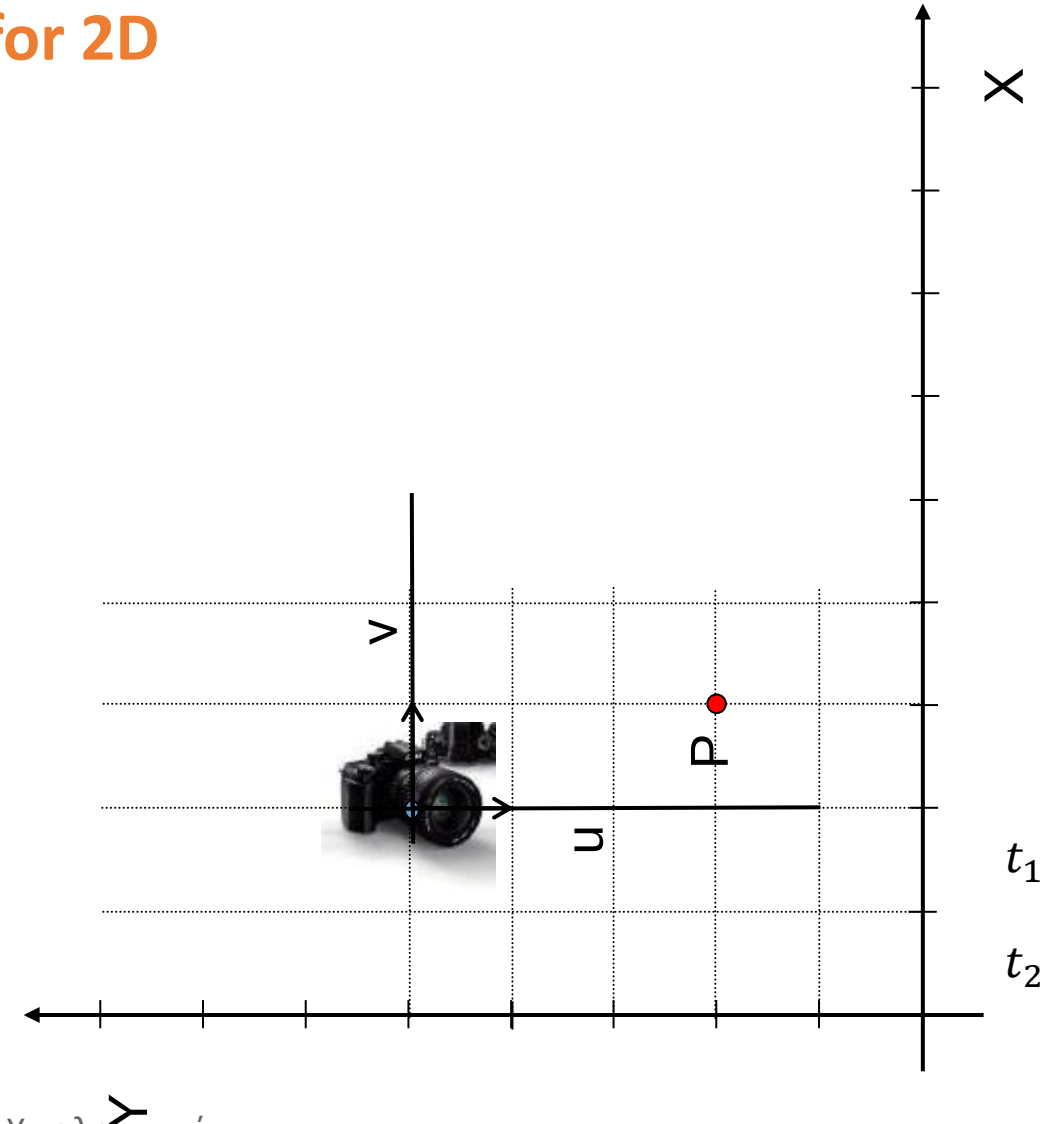
$$M = \begin{bmatrix} u_1 & u_2 & t_1 \\ v_1 & v_1 & t_2 \\ 0 & 0 & 1 \end{bmatrix}$$

$$t_1 = -\mathbf{u} \cdot VRP = -u_1 \cdot VRP_x - u_2 \cdot VRP_y$$

$$t_2 = -\mathbf{v} \cdot VRP = -v_1 \cdot VRP_x - v_2 \cdot VRP_y$$

World coordinate system (WC) and View Coordinates (VC)

An example for 2D



$$VRP = (2,5)$$

$$\mathbf{u} = (0,-1)$$

$$\mathbf{v} = (1,0)$$

$$P = (3,2) \text{ σε WC}$$

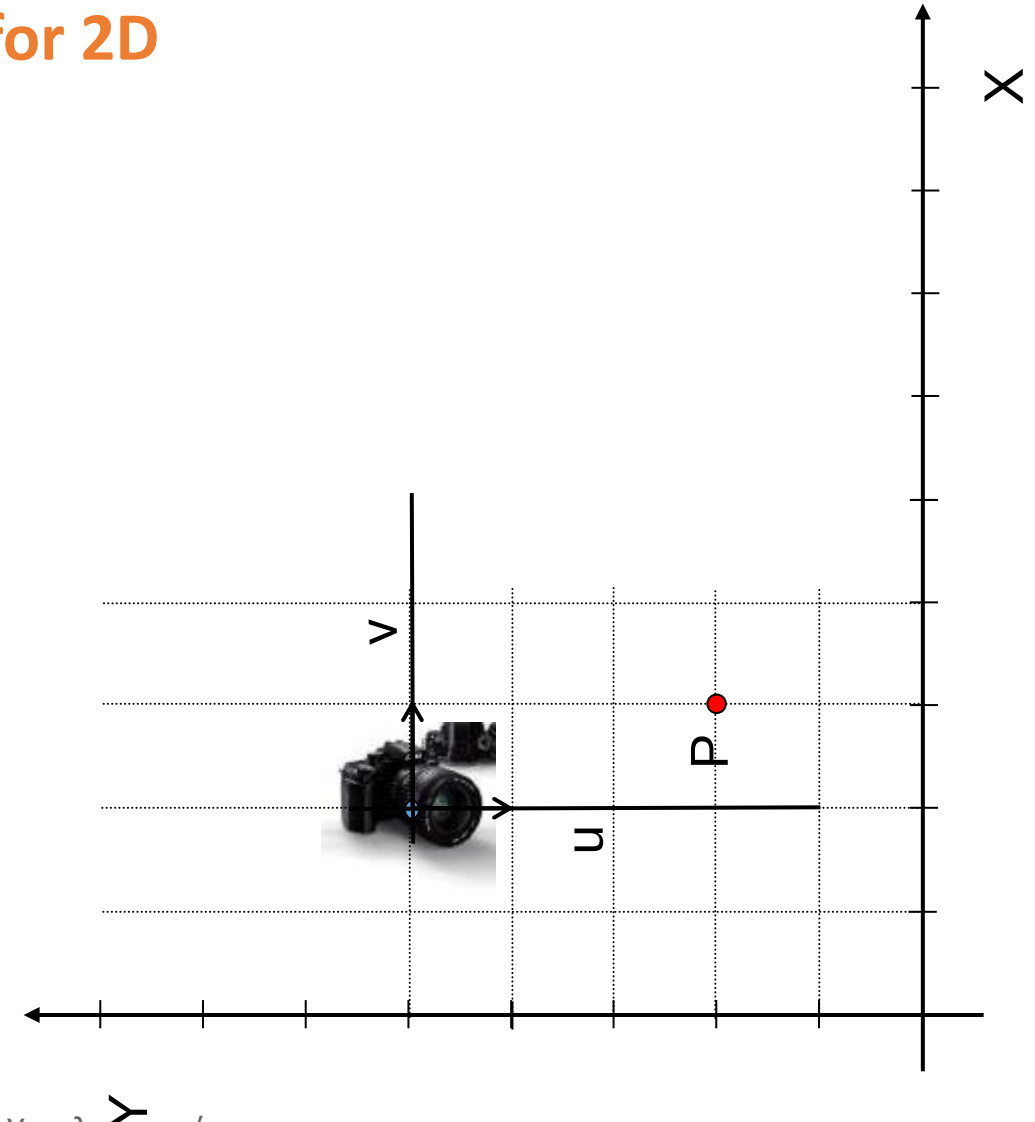
$$M = \begin{bmatrix} 0 & -1 & 5 \\ 1 & 0 & -2 \\ 0 & 0 & 1 \end{bmatrix}$$

$$t_1 = -u_1 \cdot VRP_x - u_2 \cdot VRP_y = \mathbf{0 \cdot 2 + 1 \cdot 5 = 5}$$

$$t_2 = -v_1 \cdot VRP_x - v_2 \cdot VRP_y = \mathbf{-1 \cdot 2 - 0 \cdot 5 = -2}$$

World coordinate system (WC) and View Coordinates (VC)

An example for 2D



$$VRP = (2,5)$$

$$\mathbf{u} = (0,-1)$$

$$\mathbf{v} = (1,0)$$

$P = (3,2)$ σε WC

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 5 \\ 1 & 0 & -2 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \\ 1 \end{bmatrix}$$

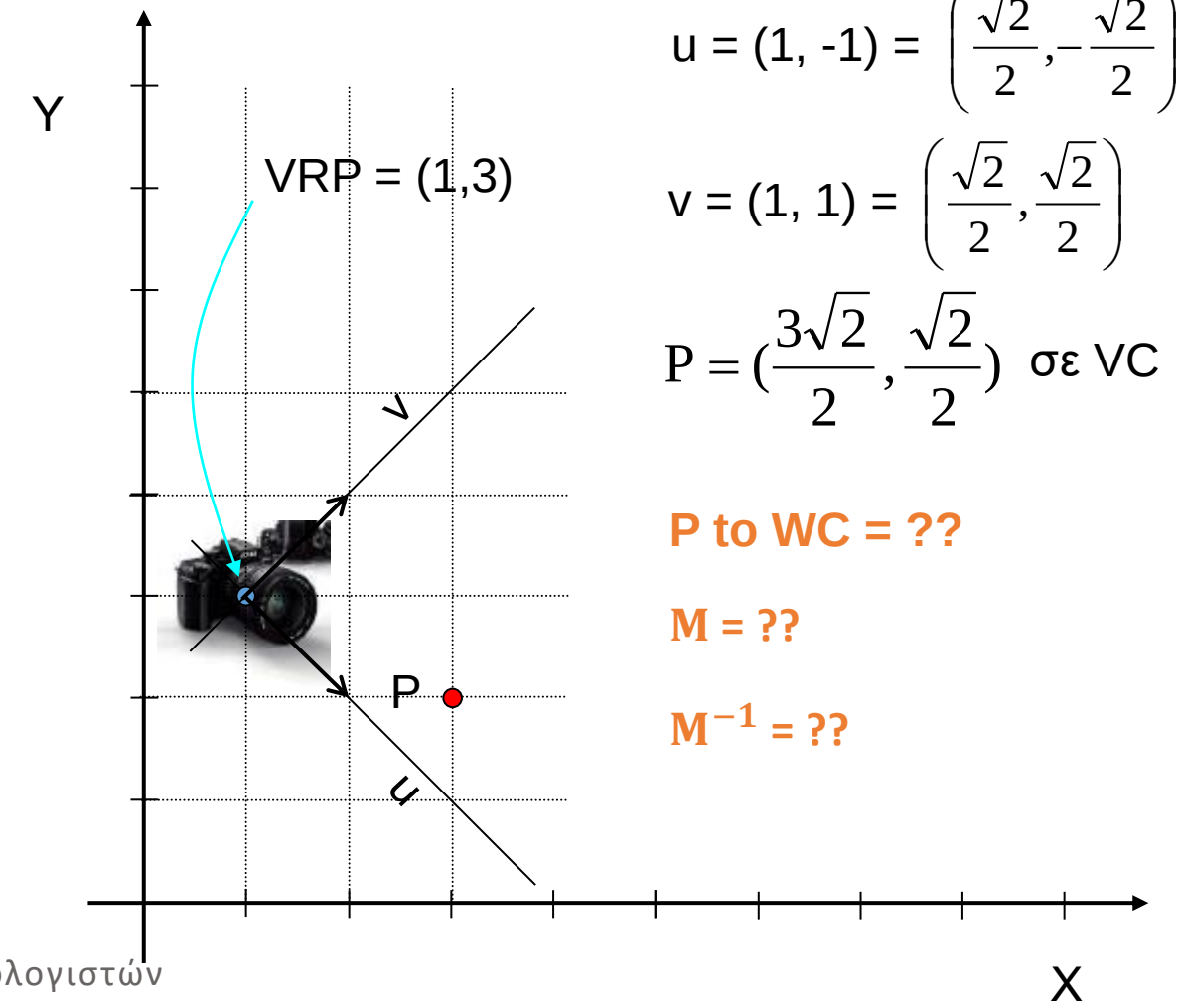
World coordinate system (WC) and View Coordinates (VC)

Transforming from VC to WC

- Sometimes we have to go from VC to WC.
  - We need the inverse, M^{-1}

World coordinate system (WC) and View Coordinates (VC)

An example for 2D



normalized vector

$u = (1, -1) = \left(\frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2}\right)$

$v = (1, 1) = \left(\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\right)$

$P = (\frac{3\sqrt{2}}{2}, \frac{\sqrt{2}}{2})$ σε VC

Data that we already know

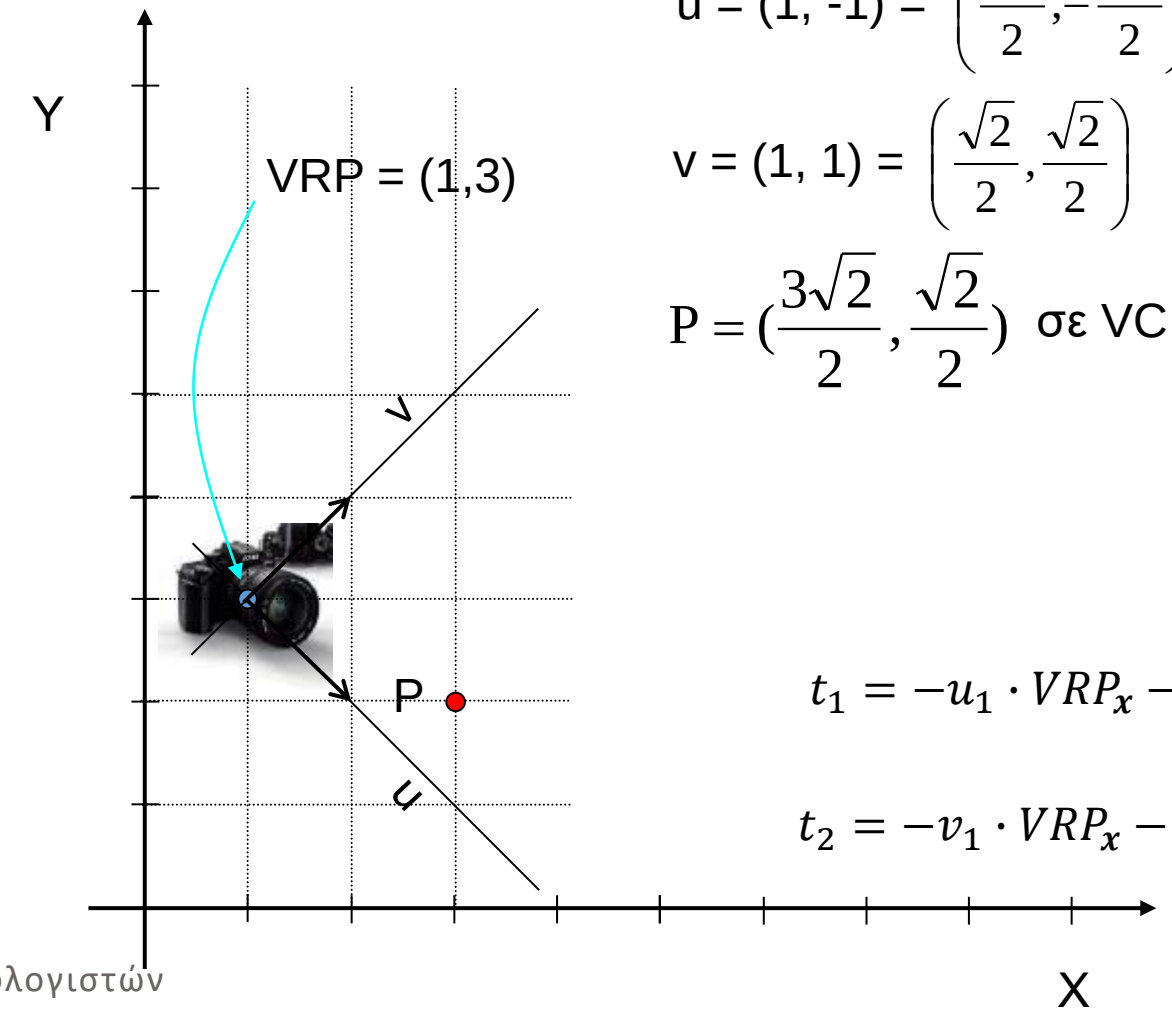
P to WC = ??

M = ??

M^{-1} = ??

World coordinate system (WC) and View Coordinates (VC)

An example for 2D



$u = (1, -1) = \left(\frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2}\right)$

$v = (1, 1) = \left(\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\right)$

$P = (\frac{3\sqrt{2}}{2}, \frac{\sqrt{2}}{2})$ σε VC

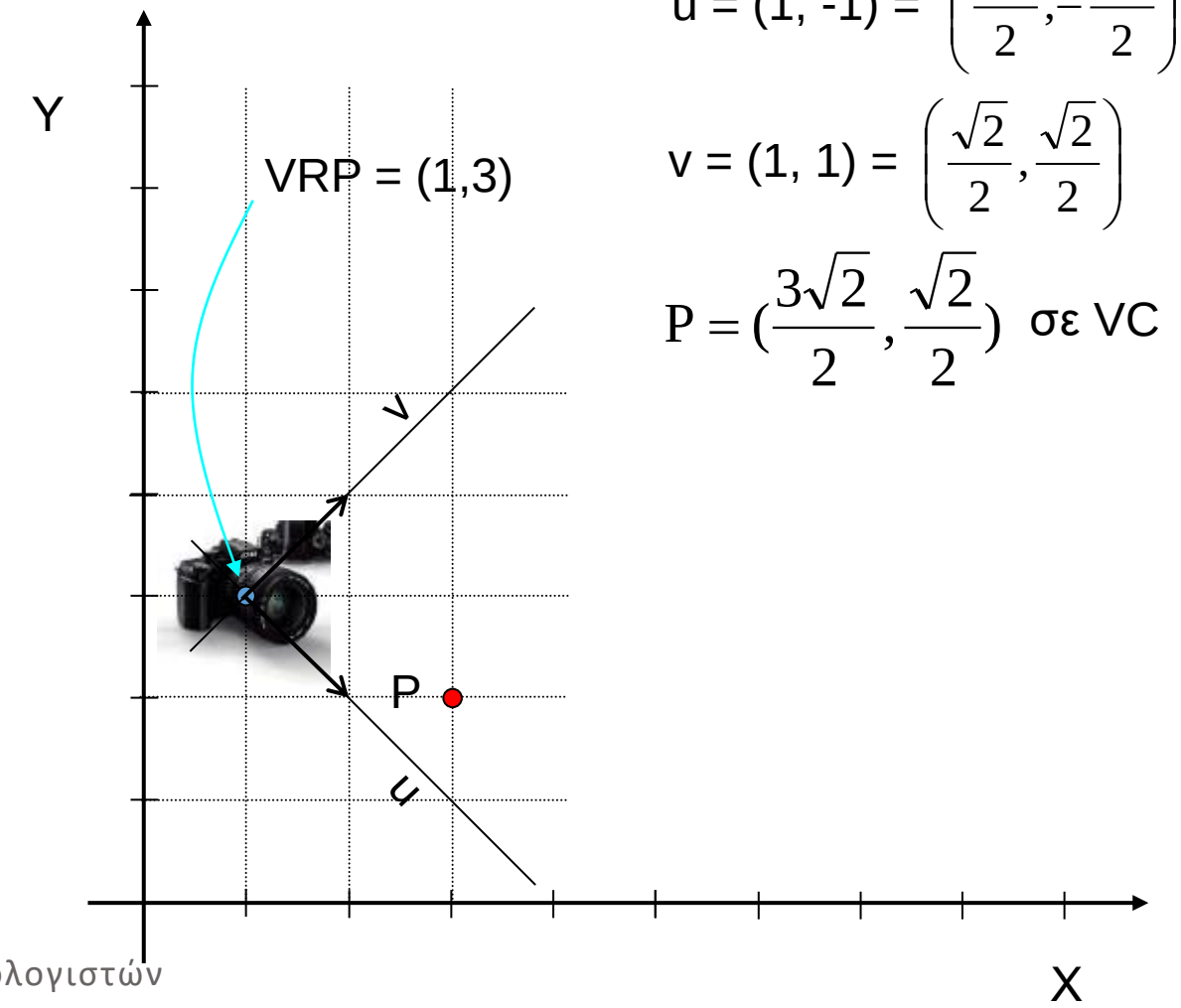
$$M = \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & \sqrt{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & -2\sqrt{2} \\ 0 & 0 & 1 \end{bmatrix}$$

$$t_1 = -u_1 \cdot VRP_x - u_2 \cdot VRP_y = -\frac{\sqrt{2}}{2} \cdot 1 + \frac{\sqrt{2}}{2} \cdot 3 = \sqrt{2}$$

$$t_2 = -v_1 \cdot VRP_x - v_2 \cdot VRP_y = -\frac{\sqrt{2}}{2} \cdot 1 - \frac{\sqrt{2}}{2} \cdot 3 = -2\sqrt{2}$$

World coordinate system (WC) and View Coordinates (VC)

An example for 2D



$u = (1, -1) = \left(\frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2}\right)$

$v = (1, 1) = \left(\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\right)$

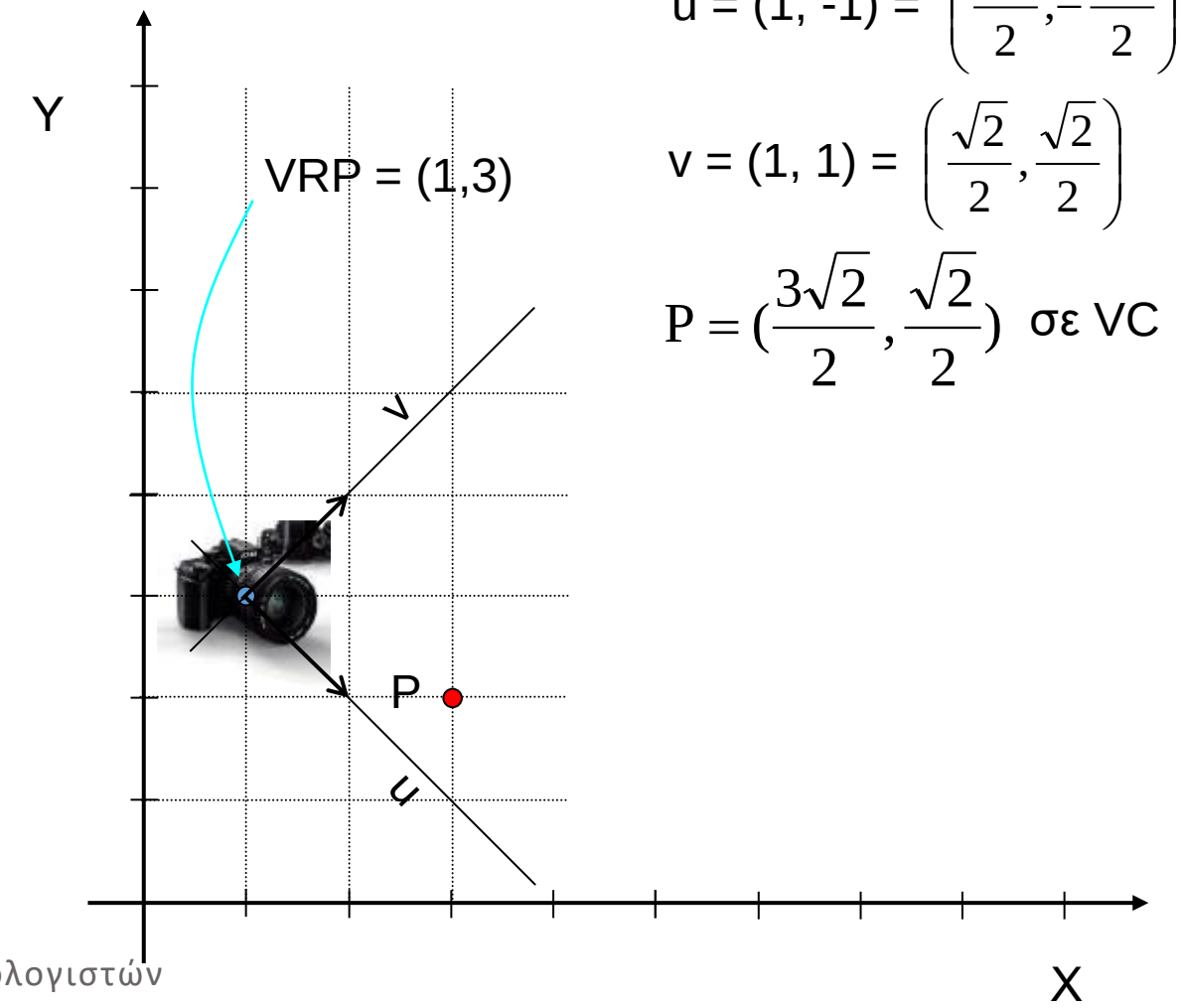
$P = (\frac{3\sqrt{2}}{2}, \frac{\sqrt{2}}{2})$ σε VC

$$M = \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & \sqrt{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & -2\sqrt{2} \\ 0 & 0 & 1 \end{bmatrix}$$

$$M^{-1} = \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 1 \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 3 \\ 0 & 0 & 1 \end{bmatrix}$$

World coordinate system (WC) and View Coordinates (VC)

An example for 2D



$u = (1, -1) = \left(\frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2}\right)$

$v = (1, 1) = \left(\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\right)$

$P = (\frac{3\sqrt{2}}{2}, \frac{\sqrt{2}}{2})$ σε VC

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 1 \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 3 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{3\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix}$$

Coordinate system

- We defined the coordinate system of the camera
- however the objects are in 3d
- **Next step:** Project the scene from 3D to 2D (view plane)