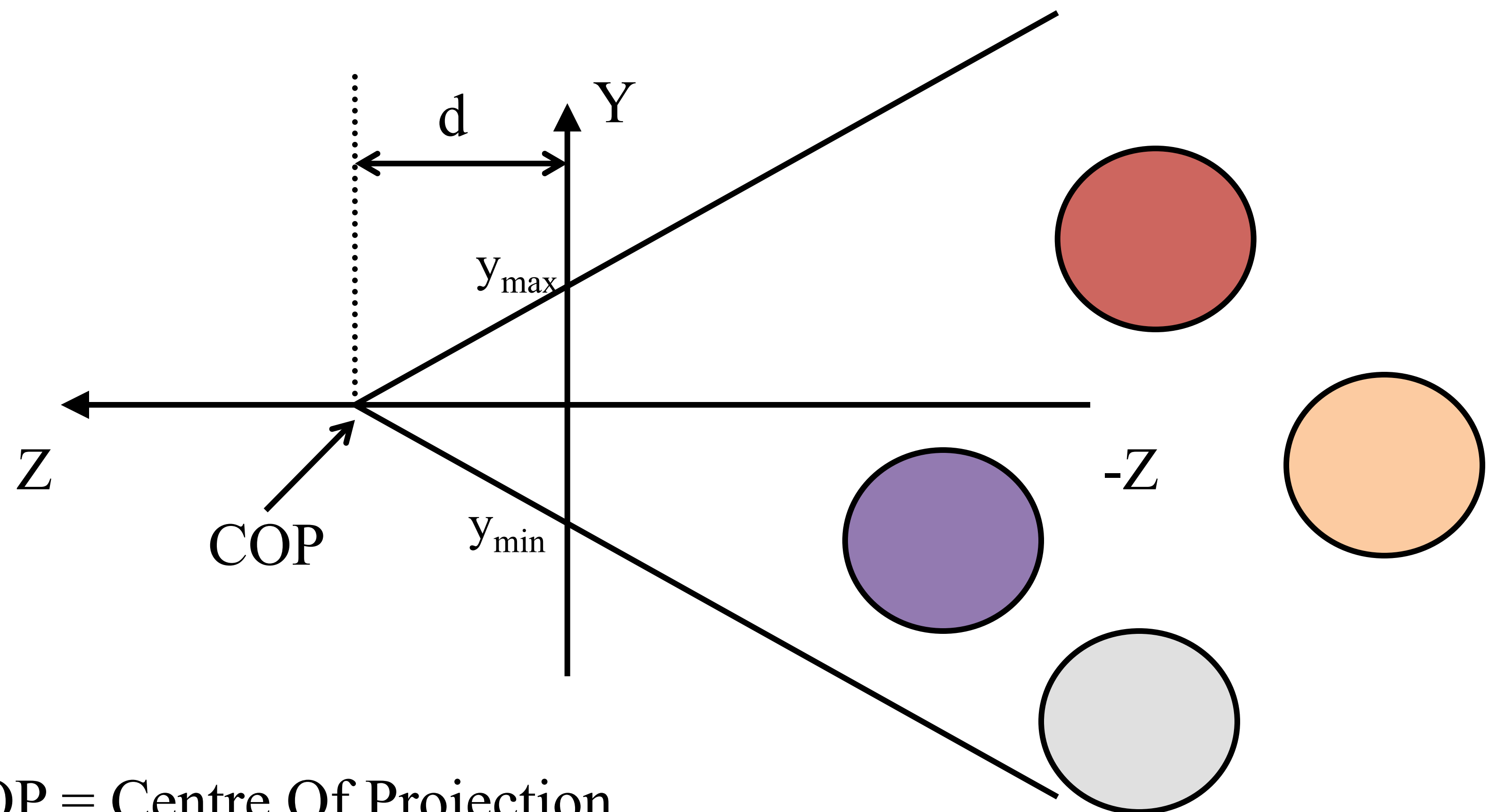
General Camera

Overview

- Simple camera is limiting: fixed in position and orientation
- We need a camera that can be moved and rotated
- We will define parameters for a camera in terms of where it “is”, the direction it points and the direction it considers to be “up” on the image

Simple Camera (Cross Section)



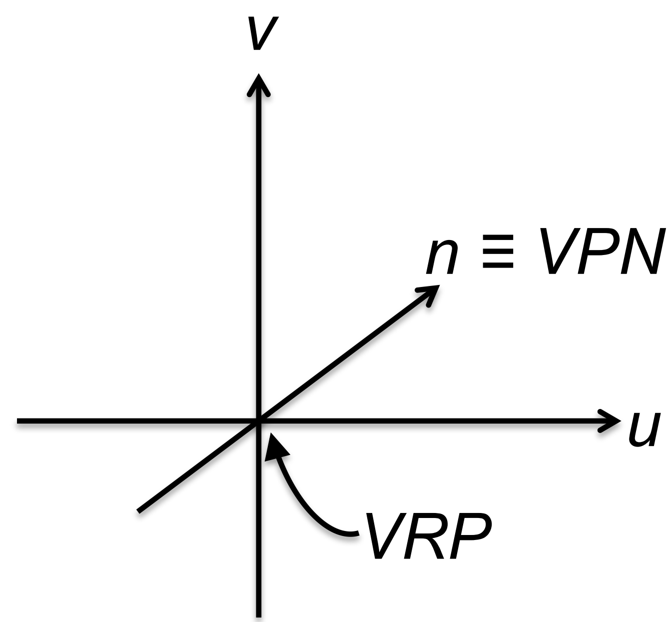
COP = Centre Of Projection

General Camera

- View Reference Point (VRP)
  - where the camera is
- View Plane Normal (VPN)
  - where the camera points
- View Up Vector (VUV)
  - which way is up to the camera
- X (or U-axis) forms LH system

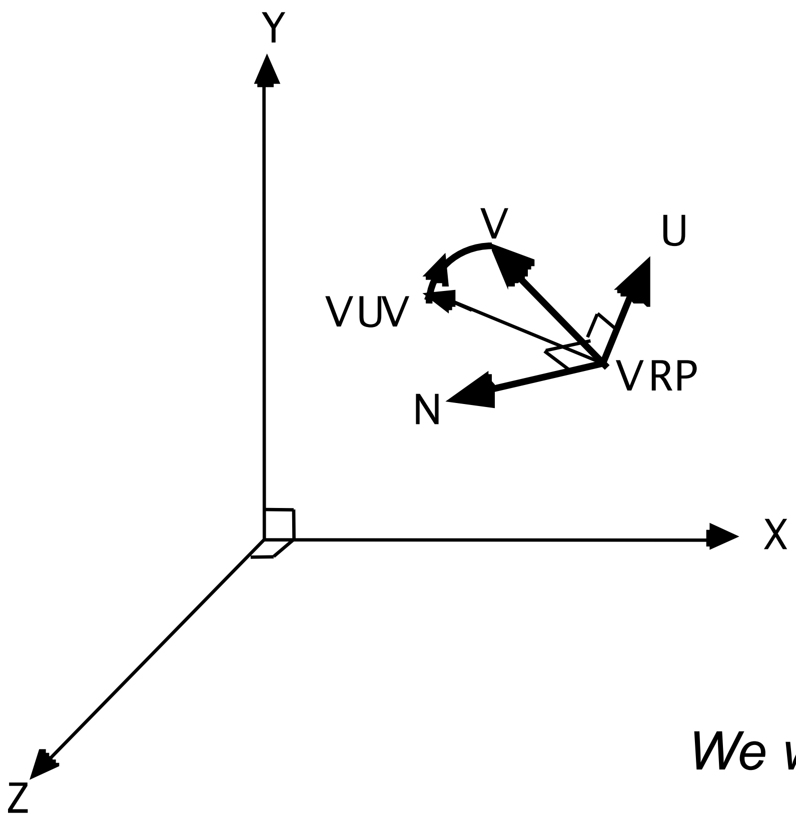
UVN Coordinates

- View Reference Point (VRP)
  - origin of VC system (VC=View Coordinates)
- View Plane Normal (VPN)
  - Z (or N-axis) of VC system
- View Up Vector (VUV)
  - determines Y (or V-axis) of VCS
- X (or U-axis) forms Left Handed system



World Coords and Viewing Coords

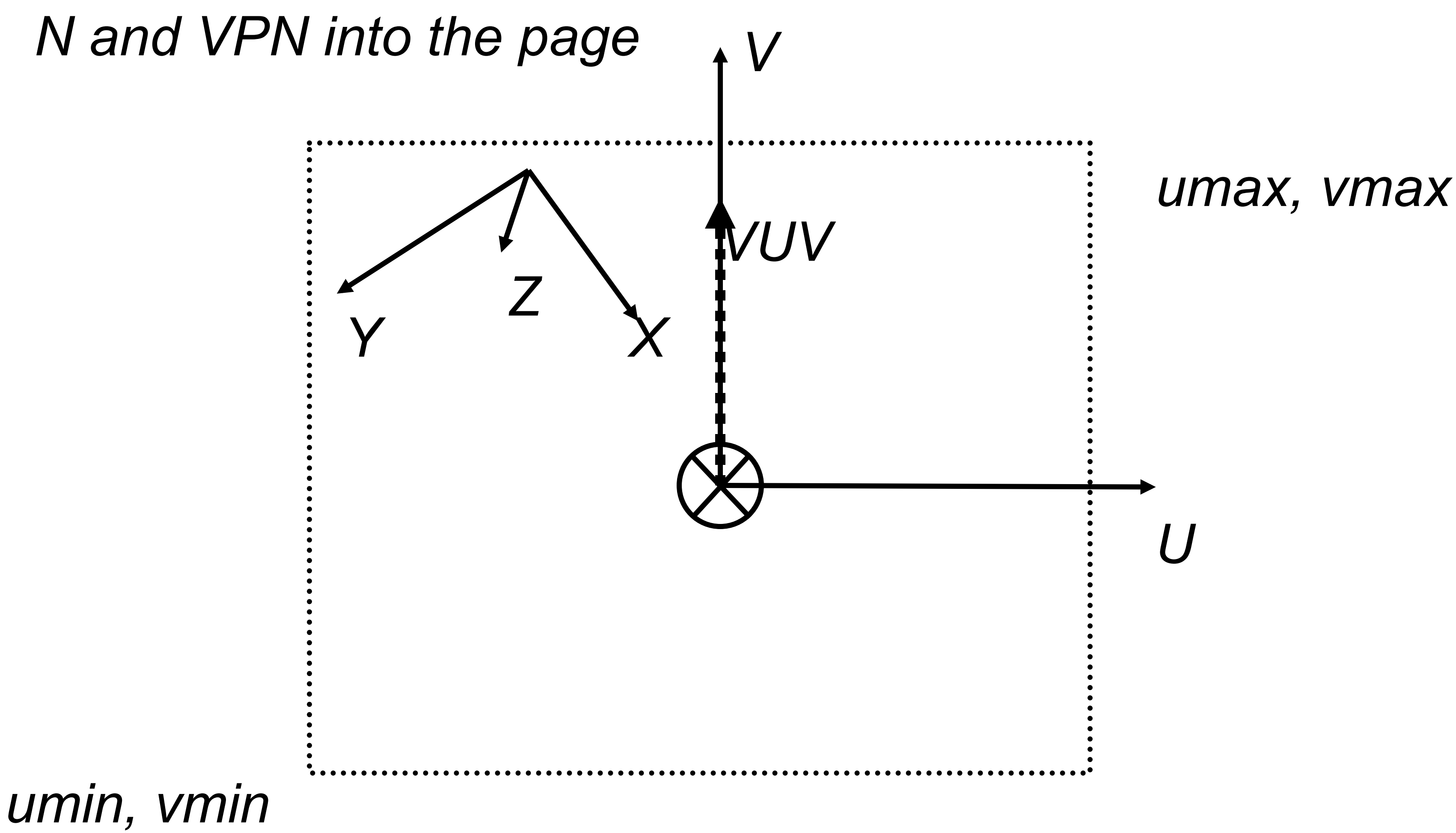
World Co-ordinates: XYZ
Viewing Co-ordinates: UVN



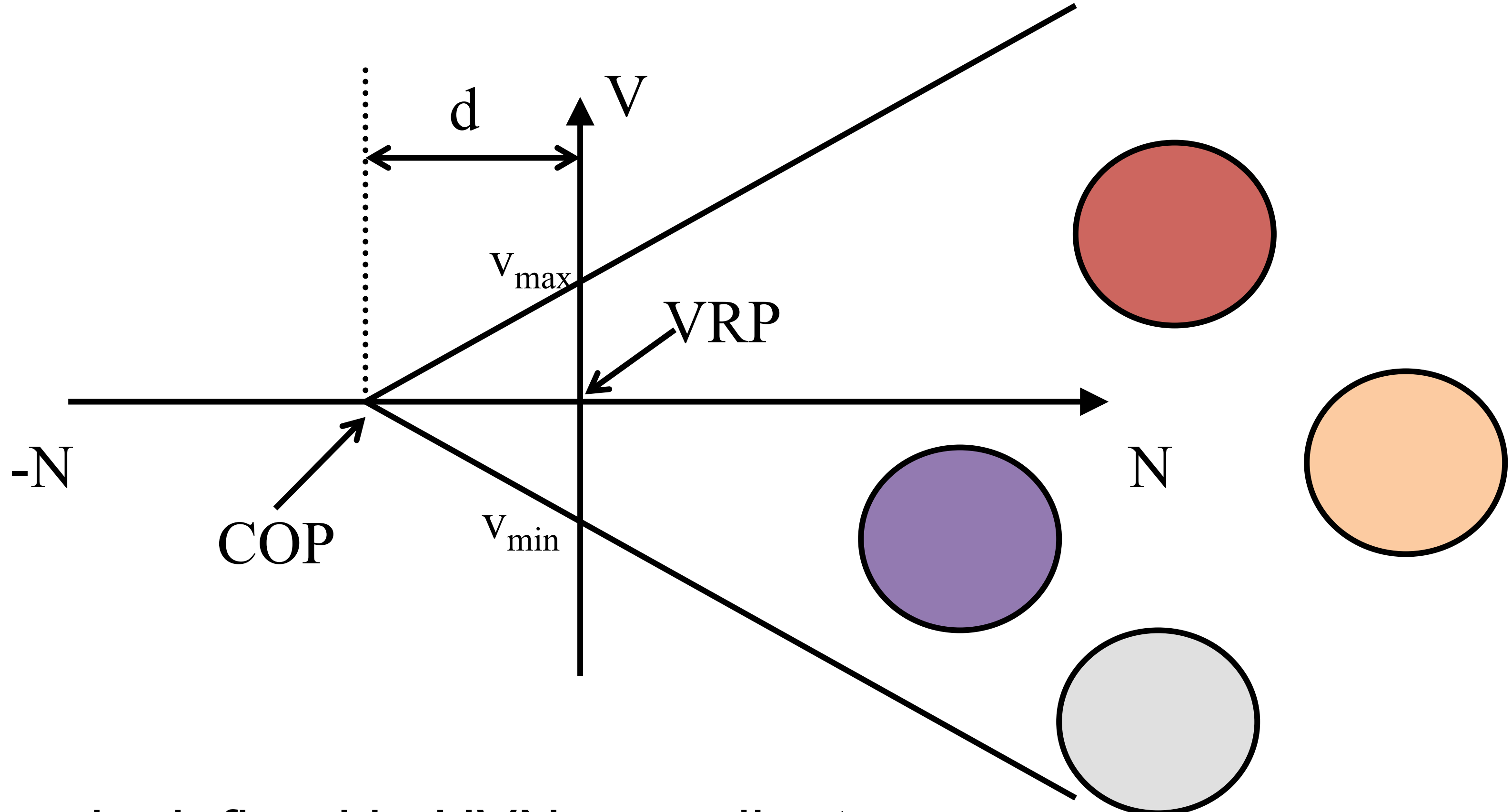
$$M = \begin{pmatrix} R_1 & R_2 & R_3 & 0 \\ R_4 & R_5 & R_6 & 0 \\ R_7 & R_8 & R_9 & 0 \\ T_1 & T_2 & T_3 & 1 \end{pmatrix}$$

We want to find a general transform of the above form that will map WC to VC

View from the Camera



Our camera becomes…



- Camera is defined in UVN co-ordinates
- …but objects in the scene will have been defined in world (XYZ) co-ordinates (this is sensible if the camera is going to move!)

Prior knowledge…

We control the placement of the camera in the scene, so we know the following (wrt world co-ordinate system):

- View Reference Point (VRP) - where the camera is
- View Plane Normal (VPN) - where the camera points
- View Up Vector (VUV) - which way is up to the camera

Finding the basis vectors

- Step 1 - find n $$n = \frac{VPN}{|VPN|}$$
- Step 2 - find u $$u = \frac{n \times VUV}{\left| n \times VUV \right|}$$
- Step 3 - find v $$v = u \times n$$

Finding the Mapping (1:Rotation)

- u,v,n must rotate under R to i,j,k of viewing space

$$\begin{pmatrix} u \\ v \\ n \end{pmatrix}\begin{pmatrix} & & \\ & R & \\ & & \end{pmatrix} = \begin{pmatrix} & & \\ & I & \\ & & \end{pmatrix}$$

- In other words:

uR = i = [1 0 0]

vR = j = [0 1 0]

nR = k = [0 0 1]

Finding the Mapping (2:Rotation)

u, v and n are *orthonormal* vectors (i.e. they are unit vectors, and are all orthogonal to each other)

$\Rightarrow$their dot products u.v, v.n, n.u are all zero

so:

$$u.v = u_1v_1 + u_2v_2 + u_3v_3 = 0$$

$$v.n = v_1n_1 + v_2n_2 + v_3n_3 = 0$$

$$n.u = n_1u_1 + n_2u_2 + n_3u_3 = 0$$

Finding the Mapping (3:Rotation)

- Also **u**, **v** and **n** are unit vectors so their magnitude is 1 thus:

  $u_1^2 + u_2^2 + u_3^2 = 1$

  $v_1^2 + v_2^2 + v_3^2 = 1$

  $n_1^2 + n_2^2 + n_3^2 = 1$

- We can exploit all this by setting R = (u^T, v^T, n^T)

$$R = \begin{pmatrix} u_1 & v_1 & n_1 \\ u_2 & v_2 & n_2 \\ u_3 & v_3 & n_3 \end{pmatrix}$$

- So $R^{-1} = R^T$

Finding the Mapping (4: Translation)

- For our equation, we call the view reference point *q*
- In uvn system *q* is (0, 0, 0, 1)

=> We want our mapping such that:

$$(q_1, \quad q_2, \quad q_3, \quad 1)\begin{vmatrix} u_1 & v_1 & n_1 & 0 \\ u_2 & v_2 & n_2 & 0 \\ u_3 & v_3 & n_3 & 0 \\ t_1 & t_2 & t_3 & 1 \end{vmatrix} = (\ 0, \quad 0, \quad 0, \quad 1)$$

Finding the Mapping (5: Translation)

So,

$$\sum_{i=1}^{3} q_i u_i + t_1 = 0 \qquad \sum_{i=1}^{3} q_i v_i + t_2 = 0 \qquad \sum_{i=1}^{3} q_i n_i + t_3 = 0$$

$$\Rightarrow \quad \begin{pmatrix} t_1 & t_2 & t_3 \end{pmatrix} = -\begin{pmatrix} \sum_{i=1}^{3} q_i u_i & \sum_{i=1}^{3} q_i v_i & \sum_{i=1}^{3} q_i n_i \end{pmatrix}$$

Complete Mapping

- Complete matrix

$$M = \begin{pmatrix} u_1 & v_1 & n_1 & 0 \\ u_2 & v_2 & n_2 & 0 \\ u_3 & v_3 & n_3 & 0 \\ -\sum_{i=1}^{3} q_i u_i & -\sum_{i=1}^{3} q_i v_i & -\sum_{i=1}^{3} q_i n_i & 1 \end{pmatrix}$$

For you to check

- If

$$M = \begin{pmatrix} R & 0 \\ -qR & 1 \end{pmatrix}$$

- Then

$$M^{-1} = \begin{pmatrix} R^T & 0 \\ q & 1 \end{pmatrix}$$

Using this for Ray-Casting

- Use a similar camera configuration (COP is usually, but not always on -n)
- To trace object must either
  - transform objects into VC
  - transform rays into WC

Ray-casting

- Transforming rays into WC
  - Find direction vectors through COP as before
  - Transform end-point once
  - Transform vector by $\begin{pmatrix} R^T & 0 \\ q & 1 \end{pmatrix}$
  - Intersect objects in WC

Ray-casting

- Transforming simple objects (e.g. spheres) into VC
  - Centre of sphere is a point so can be transformed as usual (WC to VC)
  - Radius of sphere is unchanged by rotation and translation (and spheres are spheroids if there is a non-symmetric scale)

Tradeoff

- If more rays than spheres do the former
  - transform spheres into VC
- For more complex scenes e.g. with polygons
  - transform rays into WC

Alternative Forms of the Camera

- Simple “Look At”
  - Give a VRP and a target (TP)
  - VPN = TP-VRP
  - VUV = (0 1 0) (i.e. “up” in WC)
- Field of View
  - Give horizontal and vertical FOV or one or the other and an aspect ratio
  - Calculate viewport and proceed as before

Animated Cameras

- Animate VRP (observer-cam)
- Animate VPN (look around)
- Animate TP (track-cam)
- Animate COP
  - along VPN - zoom
  - orthogonal to VPN - distort

Recap

- We created a more general camera which we can use to create views of our scenes from arbitrary positions
- Formulation of mapping from WC to VC (and back)