CS488

From 3D world into a 2D screen

Luc RENAMBOT

Transformations

- We talked about 2D and 3D transformations and how those transformations affect objects in the scene
- We discussed how polygons are usually formed into hierarchies of more meaningful objects
- We discussed how fonts are handled

3D Graphics

- We are going to talk about how we convert a set of polygons (object) in a 3D world into an image on a 2D screen

General 3D Concepts

- Taking 2D objects and mapping onto a 2D screen is pretty straightforward
- The window is the same plane as the 2D world
- Now we are taking 3D objects and mapping them onto a 2D screen

General 3D Concepts

- Here is where the advantage of separating the model world from its rendered image becomes more obvious
- The easiest way to think about converting 3D world into 2D image is the way we do it in real life - **with a camera**

General 3D Concepts

- Lets say we have an object in the real world (e.g. the Sears Tower.) The tower is a 3D object
- You can move around the tower, on the ground, on the water, in the air, and take pictures of it, converting it to a 2D image
- Depending on where you put the camera and the settings on the camera, and other factors such as light levels, you get different looking images

General 3D Concepts

- In computer graphics, we have a synthetic camera taking still or moving pictures of a synthetic environment
- While this synthetic camera gives you a much wider range of options than a real camera, you will find it is VERY easy to take a picture of nothing at all

Transformations

P(X,Y,Z)

↓

3D Object Coordinates

Modeling Transformation

↓

3D World Coordinates

Viewing Transformation

↓

3D Camera Coordinates

Projection Transformation

↓

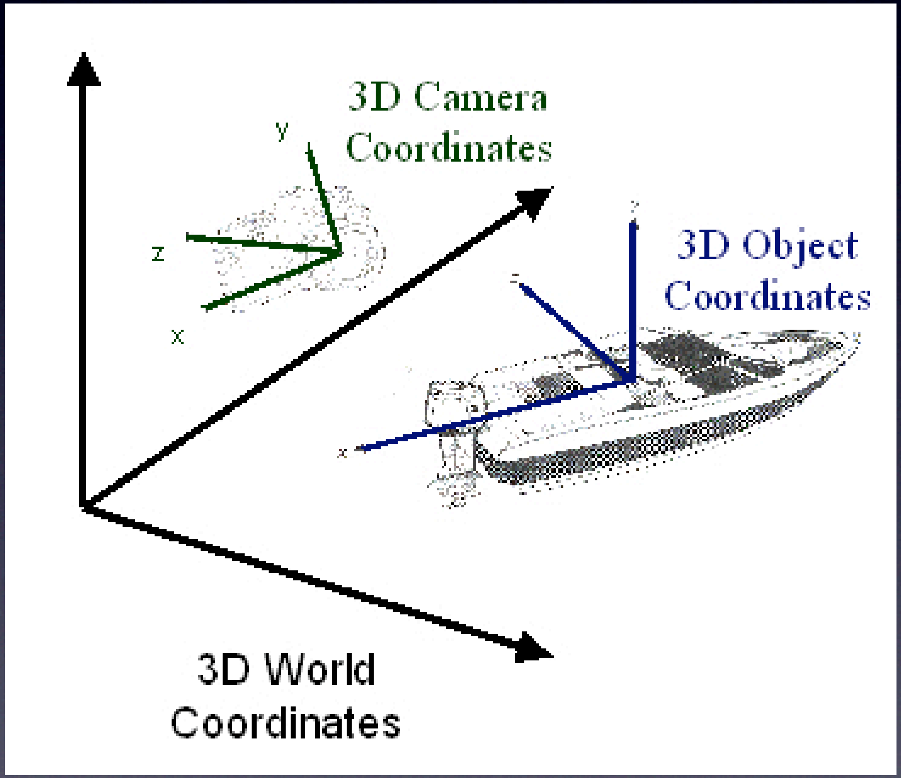
2D Screen Coordinates

Window-to-Viewport Transformation

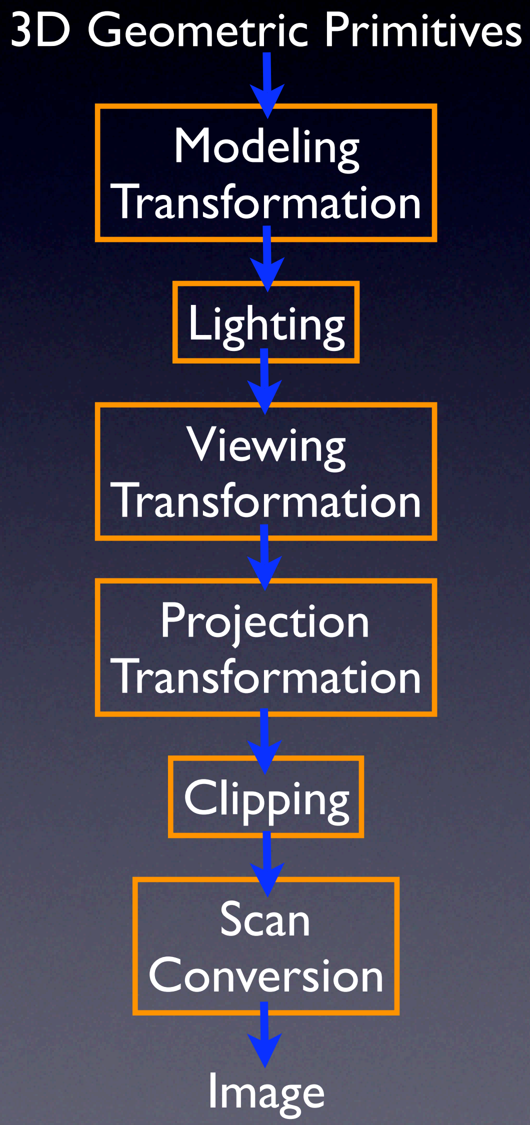
↓

2D Image Coordinates

P'(X',Y')



3D Rendering Pipeline



Transform into 3D world coordinate system

Illuminate according to lighting and reflectance

Transform into 3D camera coordinate system

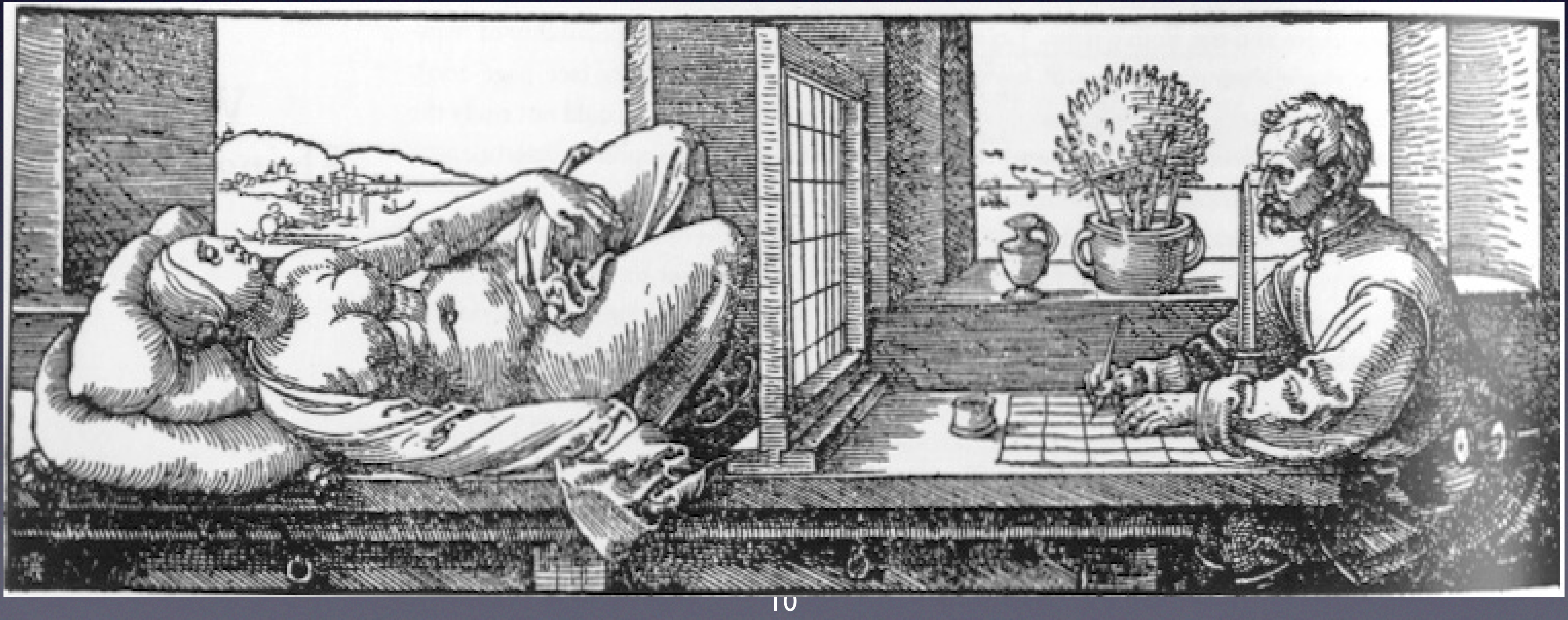
Transform into 2D camera coordinate system

Clip primitives outside camera’s view

Draw pixels (including texturing, hidden surface, etc.)

Renaissance

Albrecht Dürer, Daraughsman, Drawing a Recumbent Woman (1525) Woodcut illusion from 'The Teaching of Measurements'

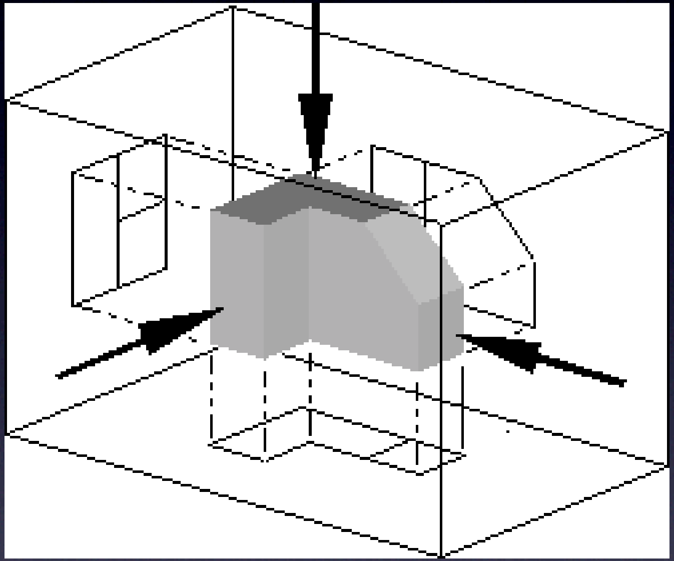


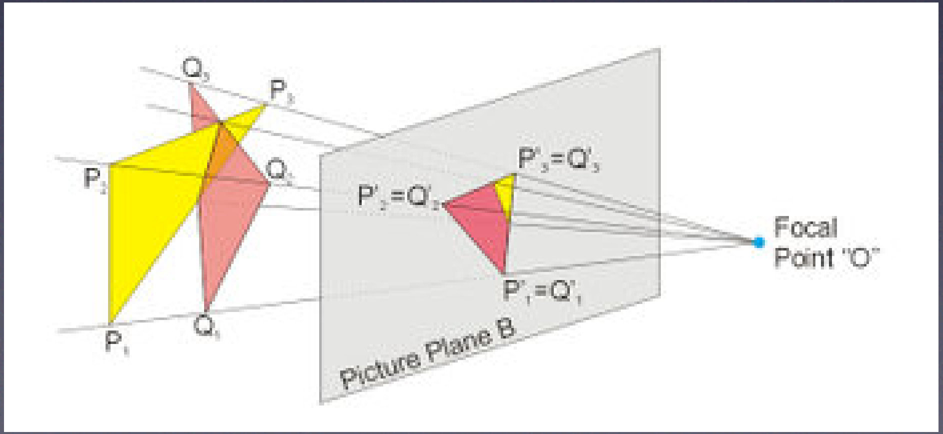
Projections

- Projection is 'formed' on the view plane (planar geometric projection)
- Rays (projectors) projected from the center of projection pass through each point of the models and intersect projection plane.
- Since everything is synthetic, the projection plane can be in front of the models, inside the models, or behind the models

Different Projections

- Two main types of projection
- Parallel projection
- Perspective projection





Parallel Projection

- Center of projection infinitely far from view plane
- Projectors will be parallel to each other
- Need to define the direction of projection (vector)
- Two sub-types
  - **Orthographic**
    - Direction of projection is normal to view plane
  - **Oblique**
    - Direction of projection not normal to view plane
- Better for drafting / CAD applications

Perspective Projection

- Center of projection finitely far from view plane
- Projectors will not be parallel to each other
- Need to define the location of the center of projection (point)
- Classified into 1, 2, or 3-point perspective
- More visually realistic
    - has perspective foreshortening (objects further away appear smaller)

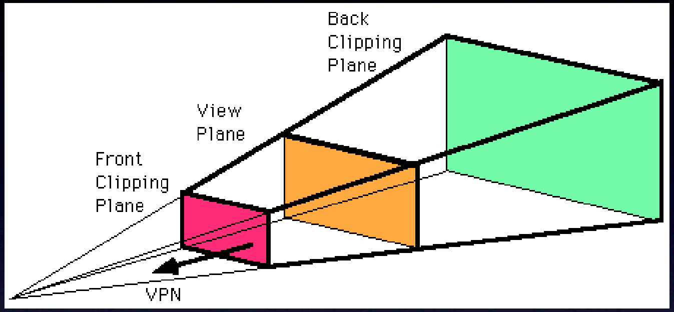
Projections

- Which type of projection is used depends on the needs of the user
- Whether the goal is the mathematically correct depiction of length and angles
- Or a realistic looking image of the object

Specifying a 3D view

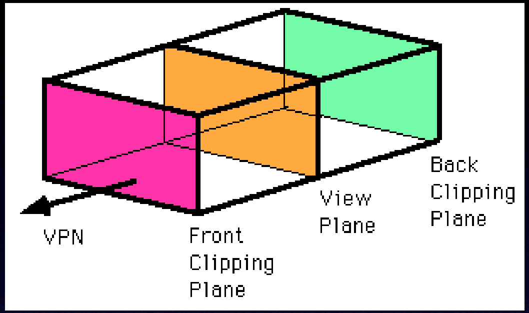
- Need to know the type of projection
- Need to know the clipping volume
- in OpenGL
  - *glFrustum*(left, right, bottom, top, near, far);
  - *glOrtho*(left, right, bottom, top, near, far);

Perspective Projection



- **glFrustum**(left, right, bottom, top, near, far);
  - Multiply by a perspective matrix
  - *left, right*
    - Specify the coordinates for the left and right vertical clipping planes
  - *bottom, top*
    - Specify the coordinates for the bottom and top horizontal clipping planes
  - *near, far*
    - Specify the distances to the near and far depth clipping planes. Both distances must be positive

Orthographic Projection



- **glOrtho(left, right, bottom, top, near, far);**
    - Multiply by a orthographic matrix
    - *left, right*
        - Specify the coordinates for the left and right vertical clipping planes
    - *bottom, top*
        - Specify the coordinates for the bottom and top horizontal clipping planes
    - *near, far*
        - Specify the distances to the nearer and farther depth clipping planes. These distances are negative if the plane is to be behind the viewer

View Plane

- Defined by
  - Point on the plane
    - View Reference Point (VRP)
  - Normal to the plane pointing towards the center of projection
    - View-Plane Normal (VPN)
- View plane can be anywhere in the world-space
- The center of projection represents the location of the viewer's eye or the camera's lens

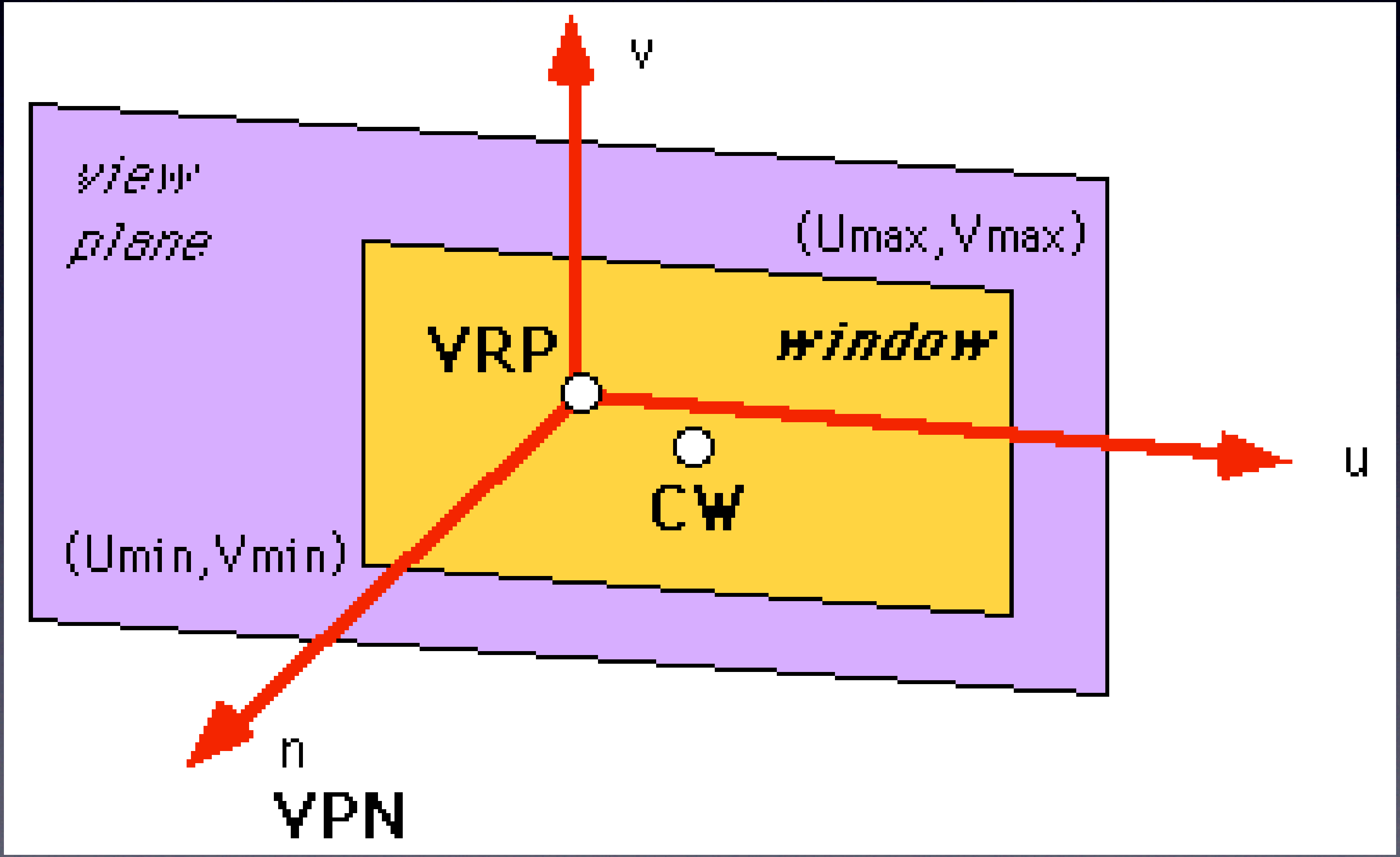
Viewing Reference Coordinate system

- Need to define a 3D Viewing Reference Coordinate system (VRC) which has axis u, v, n
  - Origin of VRC is the VRP
  - n axis of VRC is the VPN
  - v axis of VRC is called the View-UP vector (VUP)
  - u axis of VRC is defined to form a right-hand coordinate system with n and v

Window

- Since the View Plane (n=0) is infinite, we need to declare a region of that plane to be our window.
  - (Umin, Vmin) to (Umax, Vmax)
  - Center of Window (CW) on View Plane does not have to be VRP
  - VRP may not even be within the window

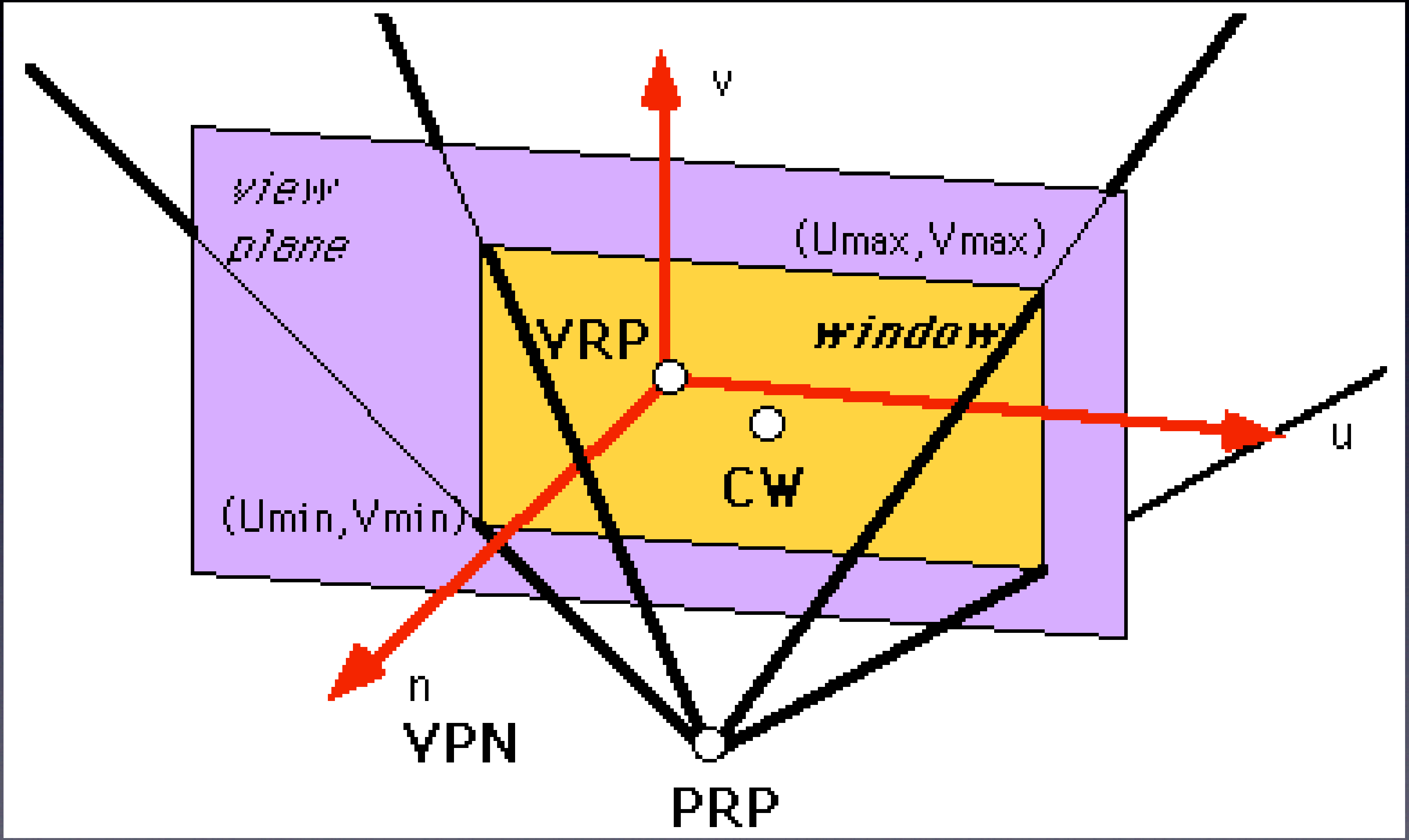
Viewing Reference Coordinate system



Projection Reference Point

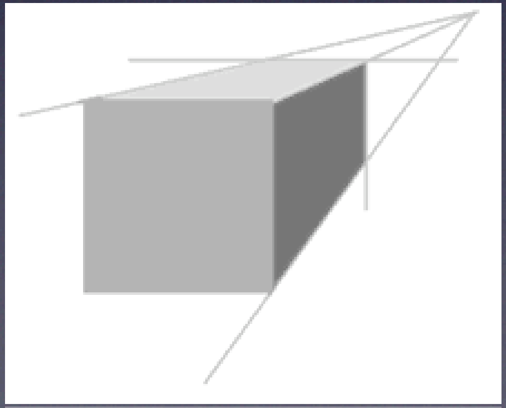
- Projection Reference Point (PRP) defines
  - Center of Projection
  - Direction of Projection (DOP)
- PRP given in VRC coordinate system (that is, its position is given relative to the VRP)
- Parallel projection
  - DOP is from PRP to CW
- Perspective projection
  - Center of Projection is the PRP
  - Line of sight is parallel to the -n axis

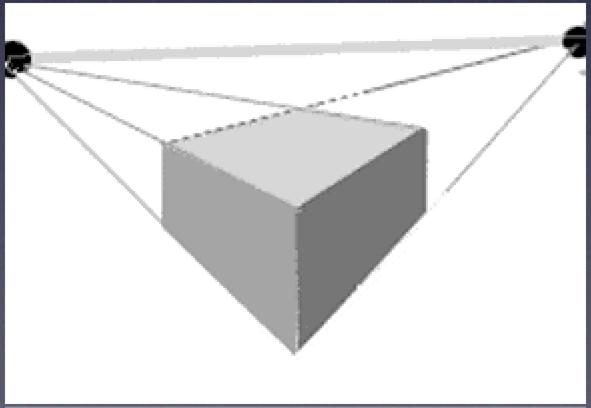
Projection Reference Point

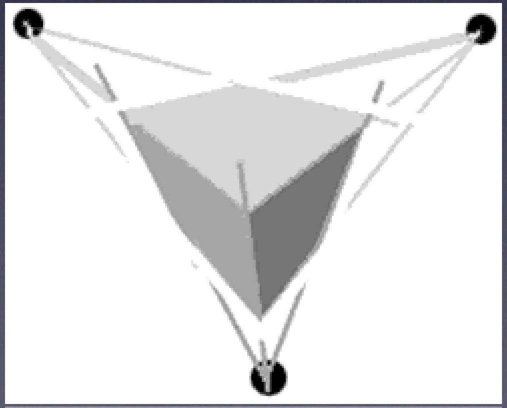


n-point Perspective

- Perspective projections categorized by the number of axis the view plane cuts
  - 1-point perspective
  - 2-point perspective
  - 3-point perspective

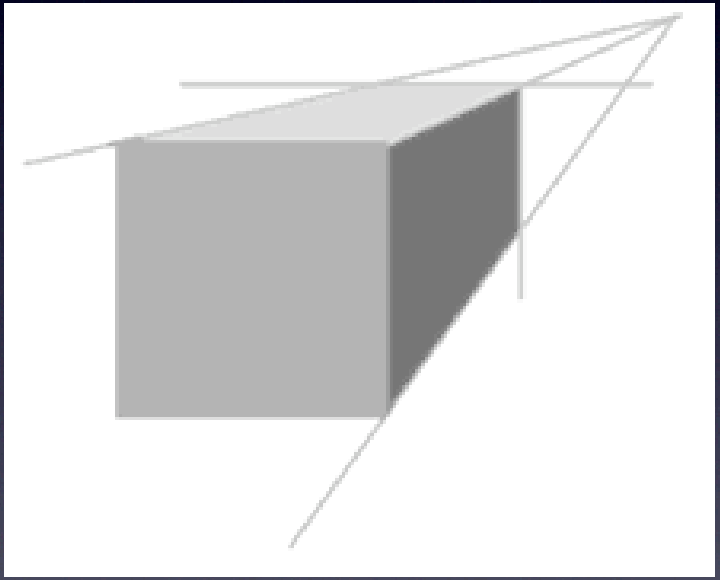

1-point


2-point


3-point

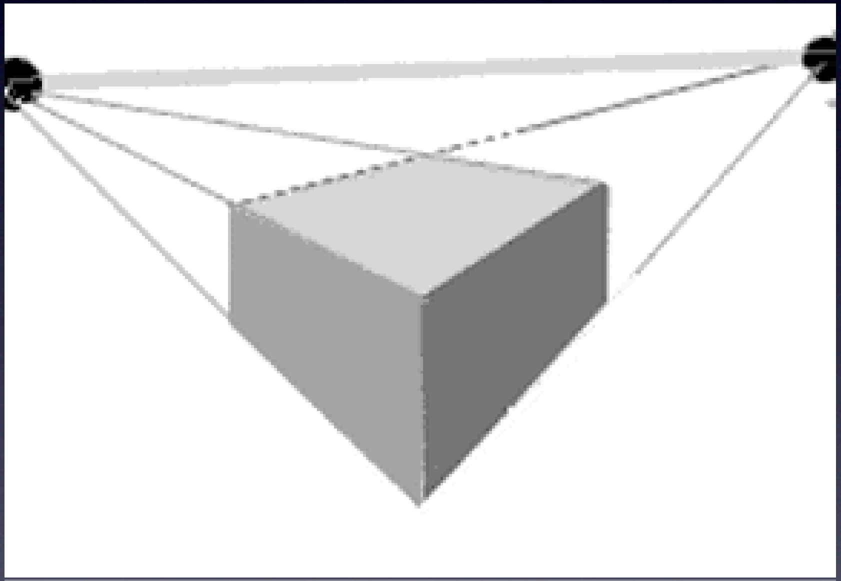
1-point Perspective

- If the view plane cuts the z axis then lines parallel to the z axis will meet at infinity
- Lines parallel to the x or y axis will not meet at infinity because they are parallel to the view plane


1-point

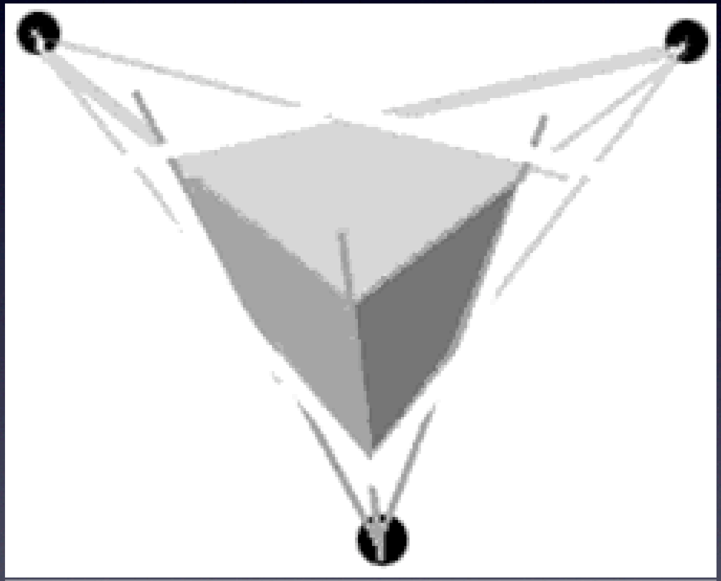
2-point Perspective

- If the view plane cuts the x and z axis then lines parallel to the x axis or the z axis will meet at infinity
- Lines parallel to the y axis will not meet at infinity because they are parallel to the view plane


2-point

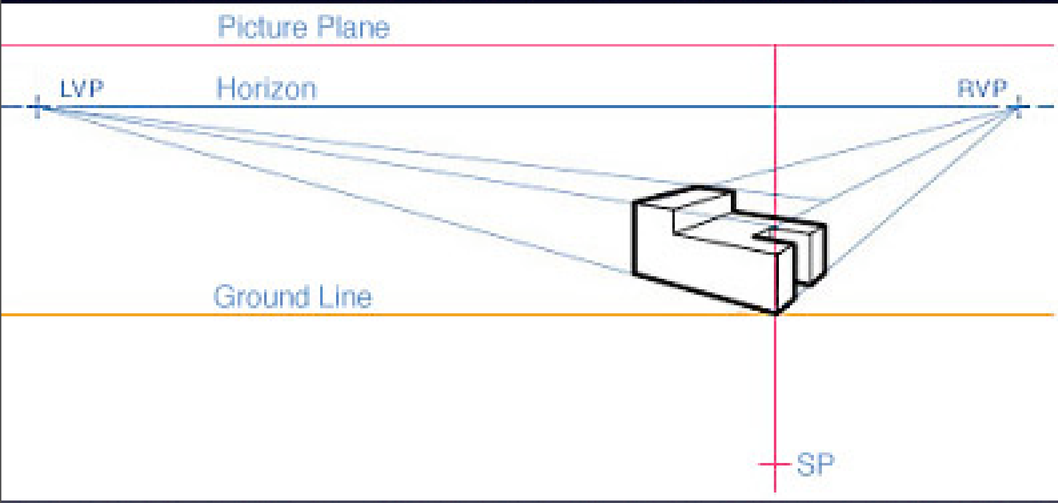
3-point Perspective

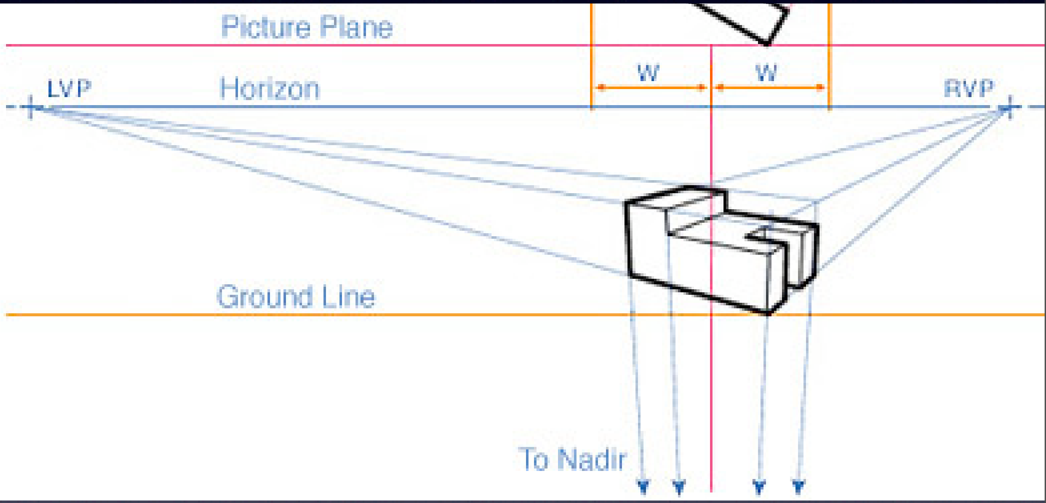
- If the view plane cuts the x, y, and z axis then lines parallel to the x, y, or z axis will meet at infinity



3-point

2-point vs 3-point





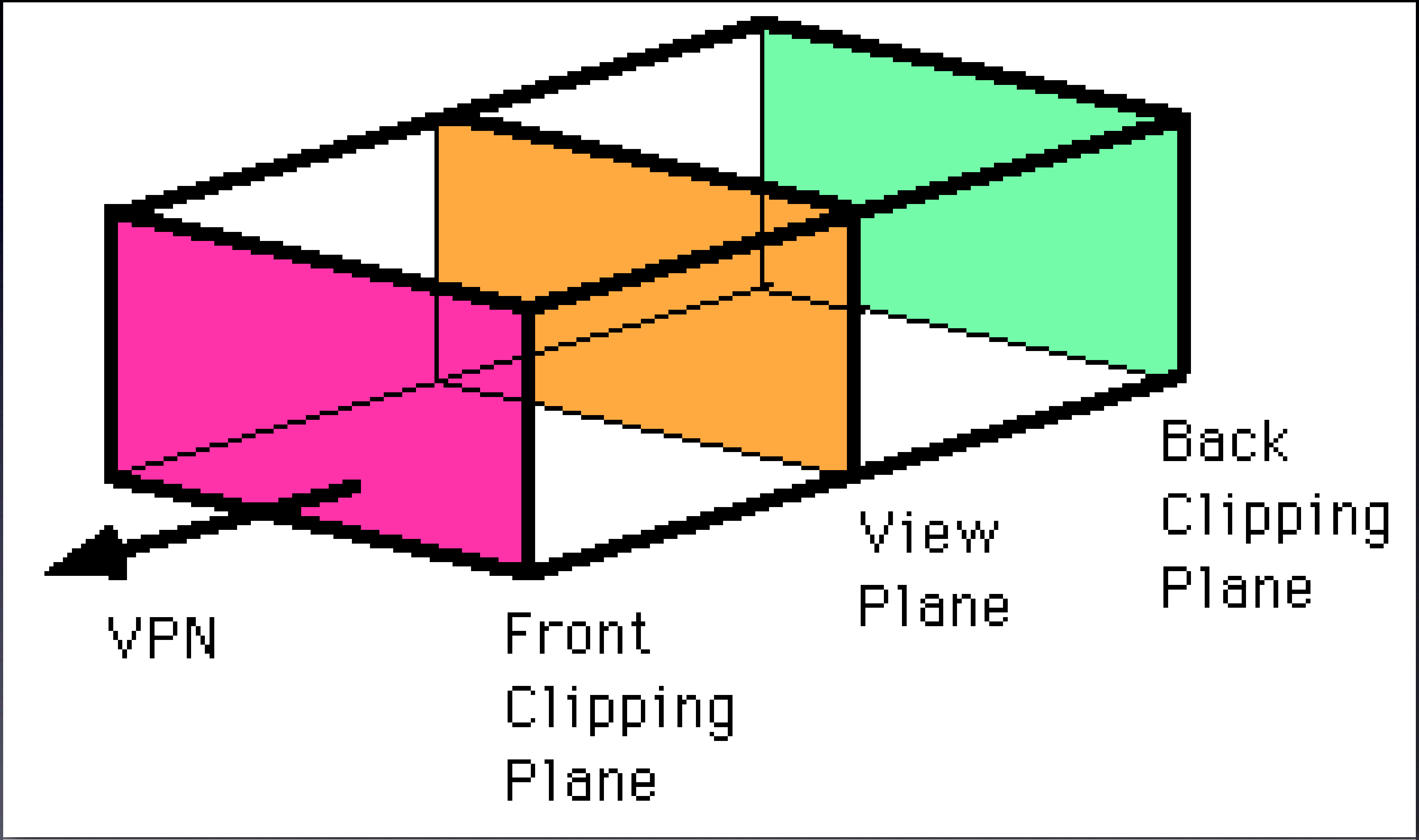
2-point

3-point

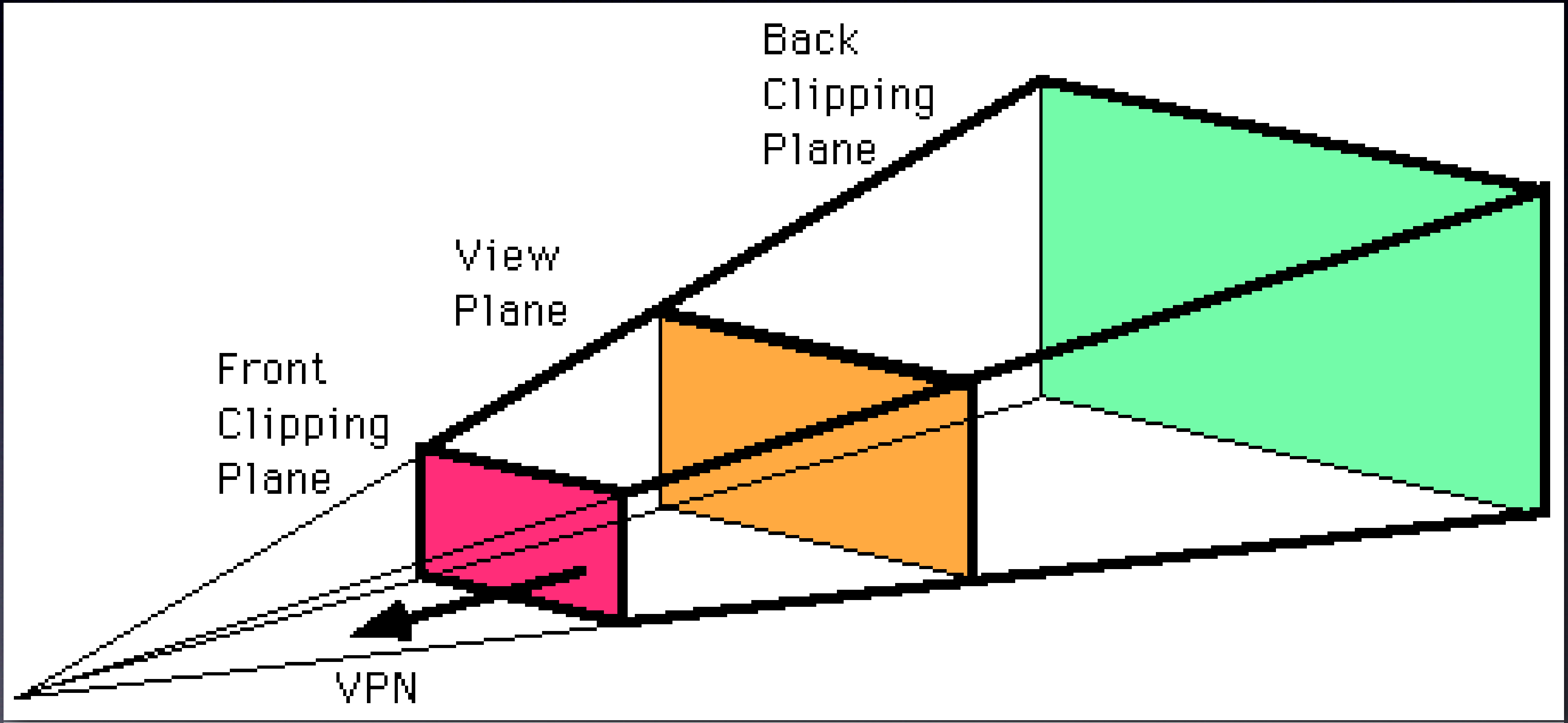
Viewing volume

- Viewing volume has 6 clipping planes
  - left, right, top, bottom, near, far
  - instead of the 4 clipping lines in the 2D case
- Perspective
  - Volume is a frustum of a 4-sided pyramid
- Orthographic
  - Volume is a rectangular parallelepiped (box)

Orthographic Projection



Perspective Projection



Parallel Examples

- In the white book, (the Foley vanDam book)
- 3 examples pages 250-252.

Perspective Examples

- Here are some examples using the same house as in the book (figure 6.24 in the white version of the Foley vanDam book)
- More pages 245-250

Example 1

0, 0,30

16, 0,30

16,10,30

8,16,30

0,10,30

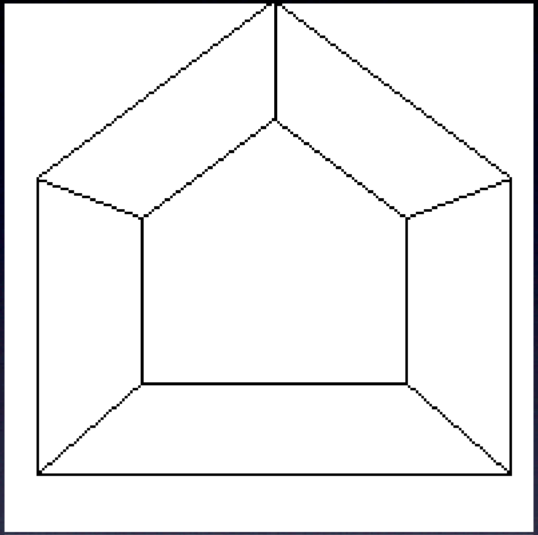
0, 0,54

16, 0,54

16,10,54

8,16,54

0,10,54



VRP= 0 0 54	or	VRP= 8 7 54
VPN= 0 0 1		VPN= 0 0 1
VUP= 0 1 0		VUP= 0 1 0
PRP= 8 7 30		PRP= 0 0 30
U: -1 to 17		U: -9 to 9
V: -2 to 16		V: -9 to 9

Example 2

0, 0,30

16, 0,30

16,10,30

8,16,30

0,10,30

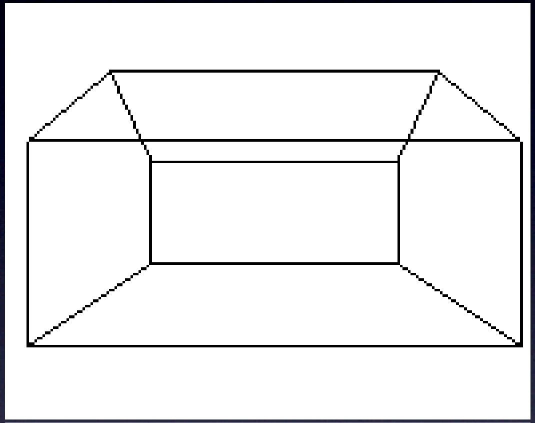
0, 0,54

16, 0,54

16,10,54

8,16,54

0,10,54



VRP= 16 0 54
VPN= 1 0 0
VUP= 0 1 0
PRP= 12 8 16

U: -1 to 25
V: -5 to 21

Example 3

0, 0,30

16, 0,30

16,10,30

8,16,30

0,10,30

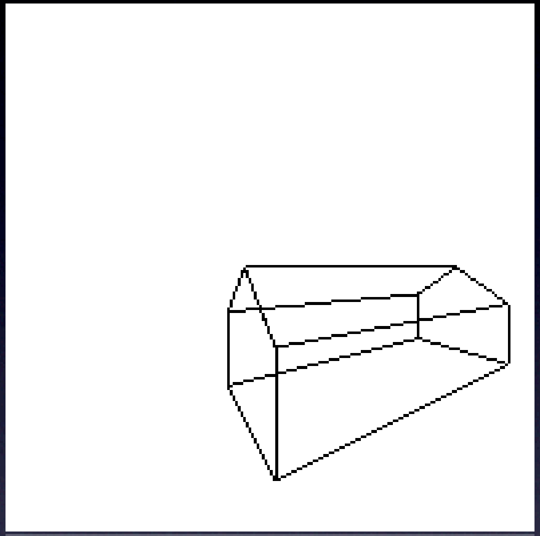
0, 0,54

16, 0,54

16,10,54

8,16,54

0,10,54



VRP= 16 0 54
VPN= 1 0 1
VUP= 0 1 0
PRP= 6 8 10

U: -22 to 22
V: -2 to 18

Finite View Volume Examples

- Adding front and back clipping planes
  - F(VRC)
  - B(VRC)
- In the white version of the Foley vanDam book see page 253

Implementation

- Lots of Matrices
  - Orthographic matrix
  - Perspective matrix
- 3D World → Normalize to the canonical view volume → Clip against canonical view volume → Project onto projection plane → Translate into viewport
- Two methods

Implementation

- Method 1
- Clipping is performed in world coordinates
    1. Extend 3D coordinates to homogeneous coordinates
    2. Normalize the homogeneous coordinates
    3. Go back to 3D coordinates
    4. Clip
    5. Extend 3D coordinates to homogeneous coordinates
    6. Perform projection
    7. Translate and Scale into device coordinates
    8. Go to 2D coordinates

Implementation

- Method 2
- Clipping is performed in homogeneous coordinates
  1. Extend 3D coordinates to homogeneous coordinates
  2. Normalize the homogeneous coordinates
  3. Clip
  4. Translate and Scale into device coordinates
  5. Go to 2D coordinates

Next Time

- Implementation...