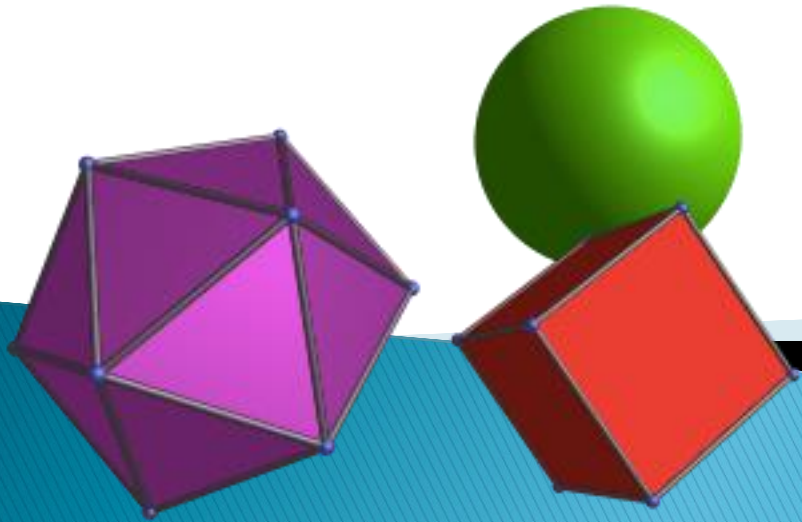
Computer Graphics ST0275

EAFIT University



Agenda

- How to represent a 3D object as a polygon mesh.
- Parametric surface
- OctTrees
- Projection of 3D objects to a 2D surface placing the camera in an arbitrary position

Motivation

- The world we see is 3D
- 3D graphics are more powerful than 2D graphics
- 3D graphics are needed for
  - Games
  - Movies
  - Virtual Reality
  - …

Representation

- In 2D, we needed two numbers to locate a point:
  - In polar coordinates: radius and angle
  - In the Cartesian plane: X and Y
- In 3D, we need three degrees of freedom (3 numbers):
  - In cylindrical coordinates: radius, angle, height
  - In spherical coordinates: radius, 2 angles
  - In the Cartesian plane: X, Y and Z

Parametric equation of the line segment

- 2D:
  - $x = x1 + t*(x2-x1)$
  - $y = y1 + t*(y2-y1)$
  - $t = [0..1]$

Parametric equation of the line segment

- 3D:
  - x = x1 + t*(x2-x1)
  - y = y1 + t*(y2-y1)
  - z = z1 + t*(z2-z1)
  - t = [0..1]

In Summary:

- Vector equation: $$\mathbf{P} = \mathbf{P_0} + t\left(\mathbf{P_1} - \mathbf{P_0}\right)$$
- In 2D
$$x = x_0 + t\left(x_1 - x_0\right)$$
$$y = y_0 + t\left(y_1 - y_0\right)$$
- In 3D
$$x = x_0 + t\left(x_1 - x_0\right)$$
$$y = y_0 + t\left(y_1 - y_0\right)$$
$$\boxed{z = z_0 + t\left(z_1 - z_0\right)}$$

Planes in 3D

- Plane equation:

$$Ax + By + Cz = D(1)$$

- If we have 3 vertices V1, V2, V3:

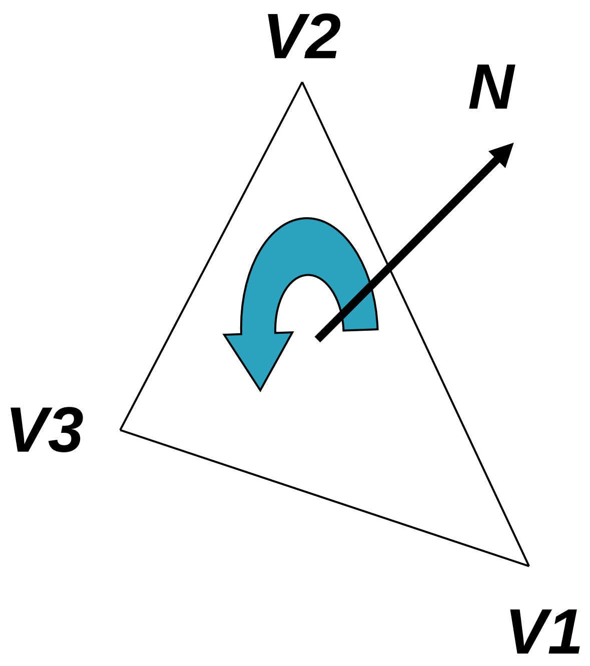
  (V2 – V1) x (V3 – V2) yields normal vector **N**

  N has the following components: A, B, C

  In order to solve for D, you take A, B, C, and the coordinates of one point in the plane into equation (1)

Relationship between a point and a plane

- V1, V2, V3 (in above equation) have to be numbered counterclockwise (looking at the polygon from the *outside*).
- If $Ax + By + Cz - D < 0$ the point is *inside*
- If $Ax + By + Cz - D > 0$ the point is *outside*



How to represent an object in 3D?

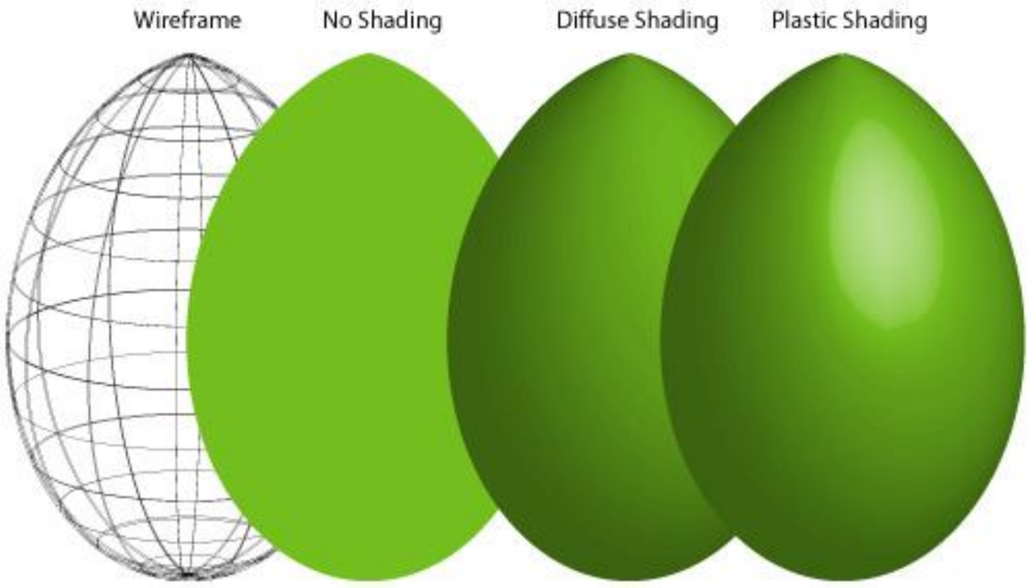
- As a polygon mesh
- Using parametric surfaces (Bézier Curves, NURBS,…)
- CSG (*Constructive Solid Geometry*)

Polygon Mesh

- The most widely used
- The object is wrapped in polygons
- In some cases textures are added

Objects as polygon meshes

- Most operations for this representation are currently done in the Graphics Card
- Once the mesh is created, the object can be rendered as:
  - "Wire-frame"
  - Solid
  - Texture mapped
  - Image from: http://z.about.com

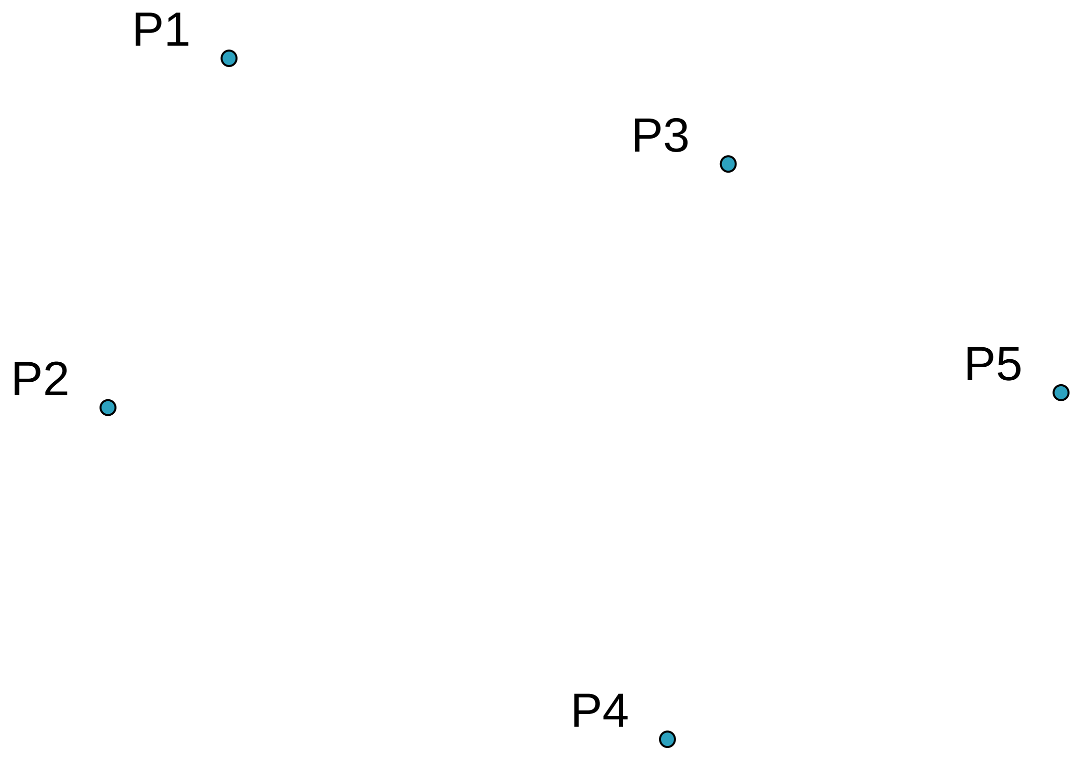


Components of the mesh

- Points
- Segments of lines
  - Two different points form a line
- Polygons
  - Three non co-linear points determine a plane

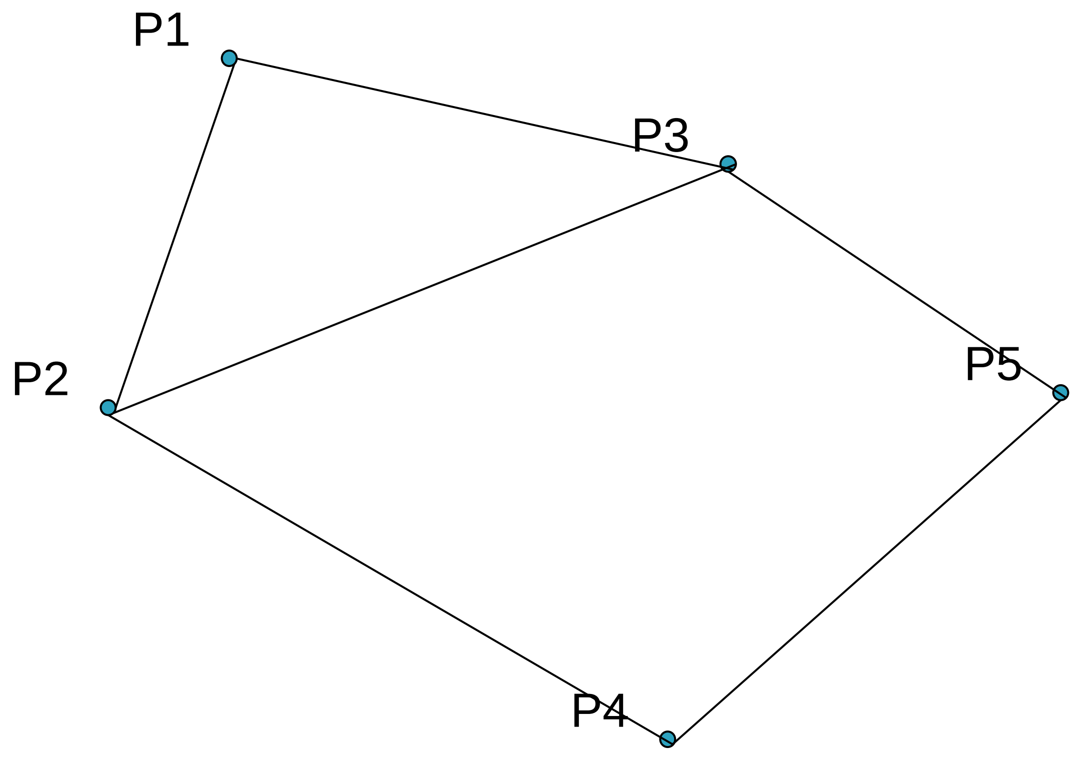
Data Structure

- Initially, define a set of points (vertices)



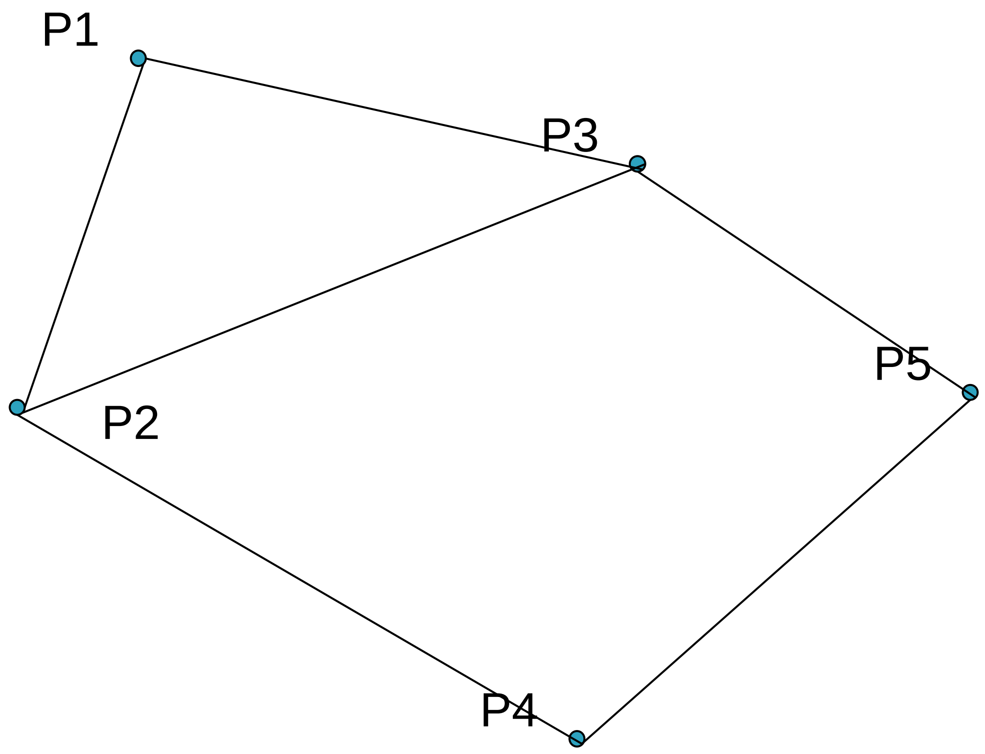
Data Structure

- Define the edges that link the vertices
  - L1 = (P1, P2)
  - L2 = (P2, P3)
  - L3 = (P3, P1)
  - L4 = (P2, P4)
  - L5 = (P4, P5)
  - L6 = (P5, P3)
  - L7 = (P3, P2)



Data Structure

- Then define the polygons
  - Edges
    - P1 = (L1, L2, L3)
    - P2 = (L4, L5, L6, L7)
  - Vertices
    - P1 = (P1, P2, P3)
    - P2 = (P2, P4, P5, P3)



Parametric Surfaces (will get back to this later)

- Generalization of the Bezier curve
- In the curve (even in space), only one parameter (u) is required.
- For the surface, two parameters are needed: *u* and *v* (why?)

$$P(u,v)=\sum_{j=0}^{m}\sum_{k=0}^{n}p_{j,k}BEZ_{j,m}(v)BEZ_{k,n}(u)$$

Parametric Surfaces

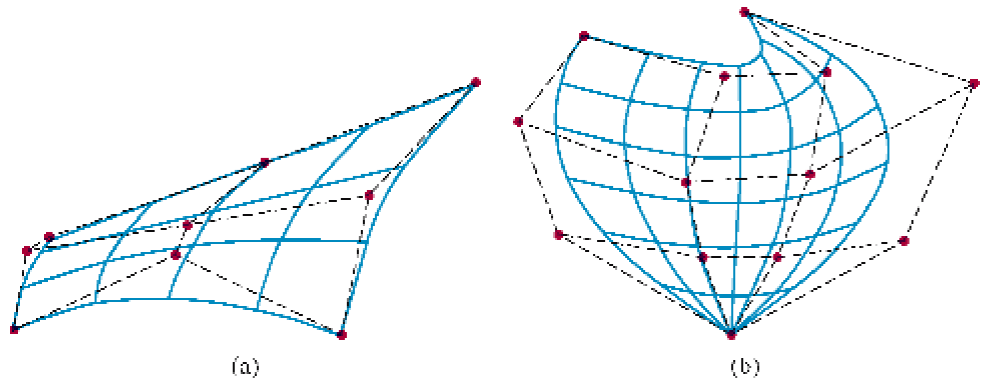


Figure 8-39

Wire-frame Bézier surfaces constructed with (a) nine control points arranged in a 3 by 3 mesh and (b) sixteen control points arranged in a 4 by 4 mesh. Dashed lines connect the control points.

OctTrees

- Octrees are volumetric representations of objects.
- Elements are represented as large volumes of “voxels” (volumetric elements).
- Each voxel has properties (color, density, …).

Quadtrees

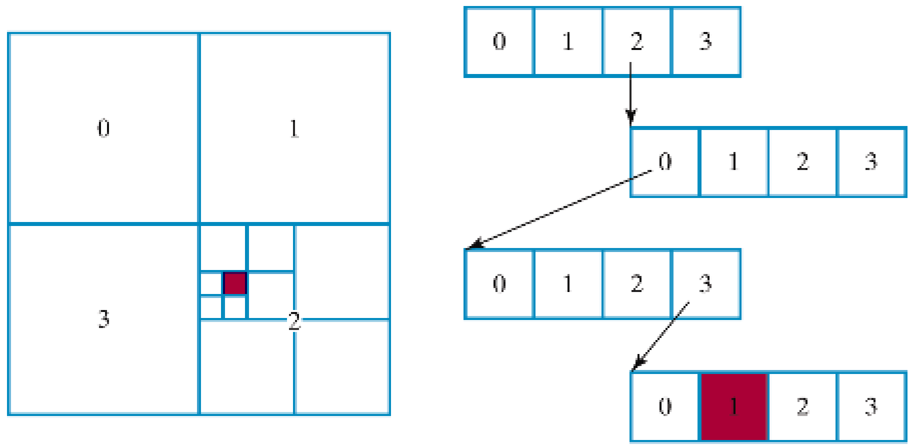


Figure 8-65

Quadtree representation for a square region of the *xy* plane that contains a single foreground-color area on a solid-color background.

Octrees

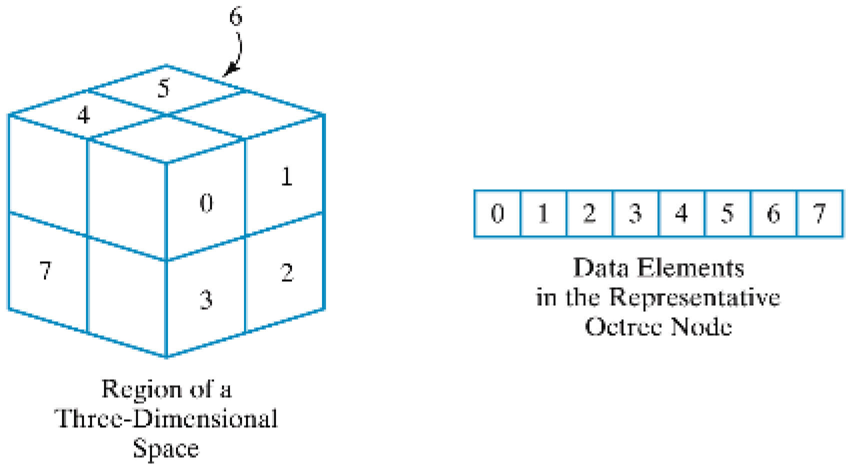

6
5
4
1
0
7
2
3
Region of a Three-Dimensional Space
0 1 2 3 4 5 6 7
Data Elements in the Representative Octree Node

Figure 8-66

A cube divided into numbered octants and the associated octree node with eight data elements.

Volume Rendering

- Visualización en 3D a partir de "cortes" 2D:

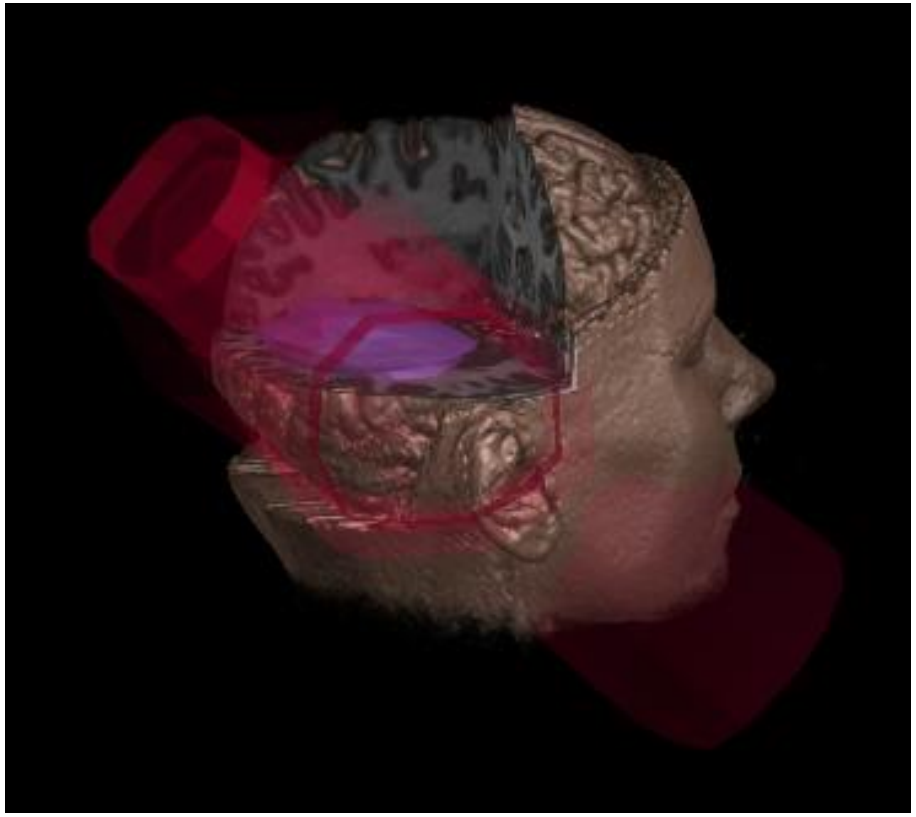
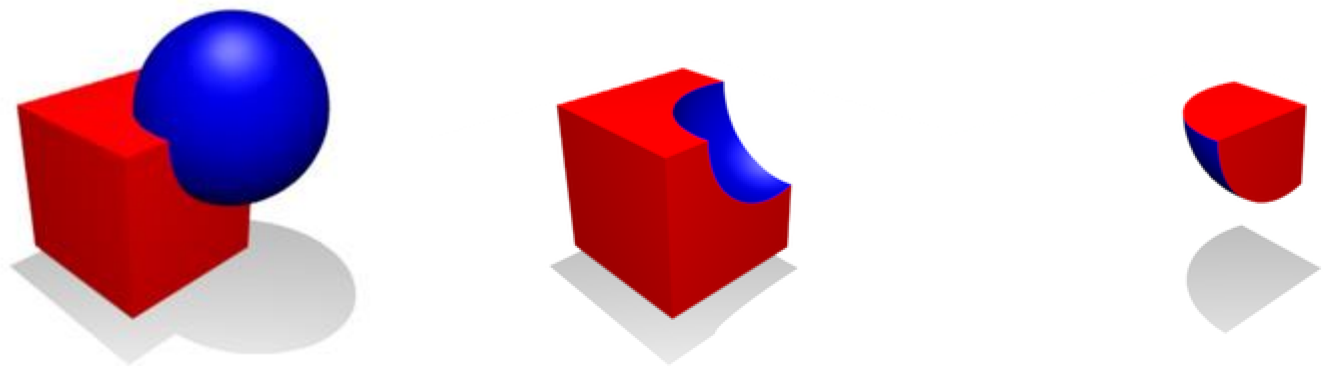


Image from: http://graphics.stanford.edu/projects/volume/

Constructive Solid Geometry

- You can create new objects from existing ones by the following operations:
  - Union
  - Intersection
  - Subtraction



Projection

- General 3D pipeline:

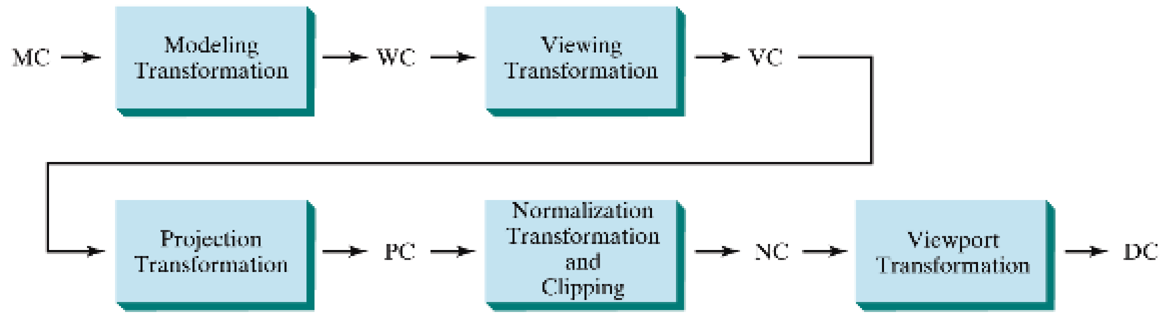
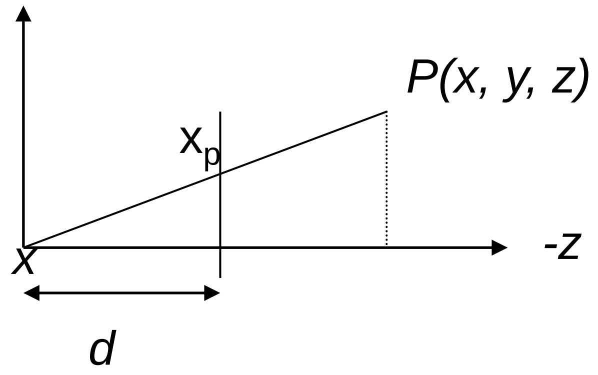


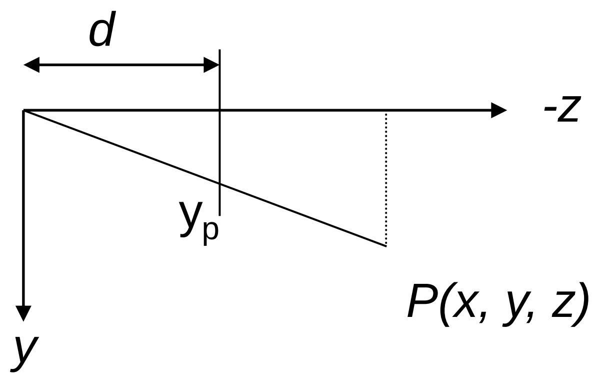
Figure 7-11

General three-dimensional transformation pipeline, from modeling coordinates to world coordinates to viewing coordinates to projection coordinates to normalized coordinates and, ultimately, to device coordinates.

Projection

- In a previous lesson, we defined the Projection Matrix
- This assumes that the observer is looking in the "–z" direction.





Projection

- We have two coordinate system:
- World and Camera

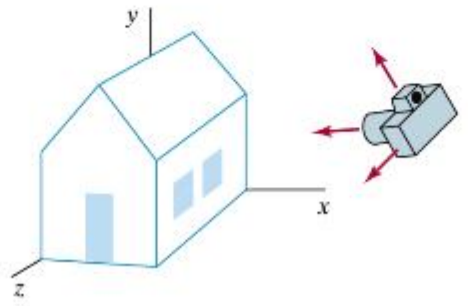


Figure 7-10

Photographing a scene involves selection of the camera position and orientation.

Projection

- Question: How to express the object in terms of the camera system?
- Answer: Transform the camera system so that it coincides with the world system.

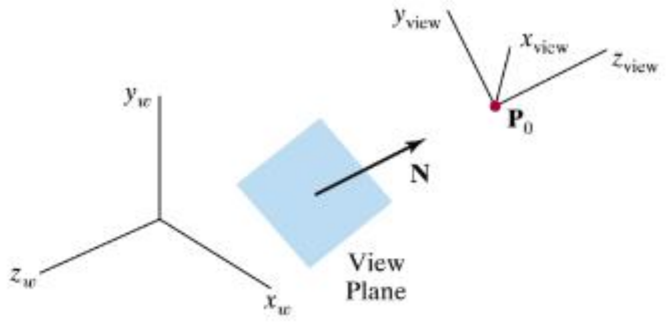


Figure 7-13

Orientation of the view plane and view-plane normal vector **N**.

Projection

- For the virtual camera we need:
  - Position
  - Orientation
    - Looking-at vector (negative, actually, to make it a right-handed system) (we will call it *n*)
    - "Up" vector (we will call it *V* (note the capital))
- Think of a photography camera
- **uvn** is called an ortho-normal base.

Projection

- 1 Translate the *camera* and *the objects in the scene* coordinate system (C.S.) to the *world* C.S.

$$P_0 = (x_0, y_{0,} z_0)$$

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

Projection

- 2 Define a right-handed set of axes that serve as camera C.S.
- *n*: view plane normal vector (similar to **Z**)
- *V*: non-orthogonal "up" vector (similar to **Y**)
- *v*: orthogonal up vector
- *u*: orthogonal "*right*" vector (similar to **X**)

$$n = \frac{N}{|N|} = (n_x, n_y, n_z)$$

$$u = \frac{V \times n}{|V \times n|} = (u_x, u_y, u_z)$$

$$v = n \times u = (v_x, v_y, v_z)$$

Projection

- Notice that

$$(n_x, n_y, n_z)$$

- Are the cosine-direction angles of unitary vector *n*.
- Same for vectors *v* and *u*.

Projection

- 3. So, the following matrix:

$$R = \begin{bmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ n_x & n_y & n_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Will rotate the camera c.s. to coincide with the world c.s.

Projection

- The coordinate transformation matrix is:

$$M = R \times T$$

$$M = \begin{bmatrix} u_x & u_y & u_z & -u \times P_0 \\ v_x & v_y & v_z & -v \times P_0 \\ n_x & n_y & n_z & -n \times P_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Projection

- And then we can use the previous projection matrix

$$M_{per} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix}$$

$$\begin{bmatrix} X & Y & Z & W \end{bmatrix}^T = \begin{bmatrix} x & y & z & z/d \end{bmatrix}^T$$

Example

- Assume the camera is looking from <0,0,0> into the positive X direction. "Up" vector is <0,1,0>. Form the **uvn** orthonormal base.
- Transform point <1,1,1> to the new base.

Retos

- Clase
  - Observador en <0, 0, 1>, mirando hacia el punto <-3, 0, 1>. Vector up: <0, 1, 0>. Hay un punto en <-3, 3, -2>. Al aplicar la transformación UVN, ¿Dónde queda?
- Casa
  - Poner la cámara en un punto cualquiera de la escena, mirando hacia el centro de la casita. Aplicar la transformación **uvn**, luego la proyección y dibujarla.

Credits

- Graphs taken from Hearn & Baker, Computer Graphics with OpenGL, 3rd Edition, Chapter 8
- First image from: http://eusebeia.dyndns.org/4d/vis/02-analogy
- Reading: Hearn & Baker, sections 8-1, 8-21, 8-22.