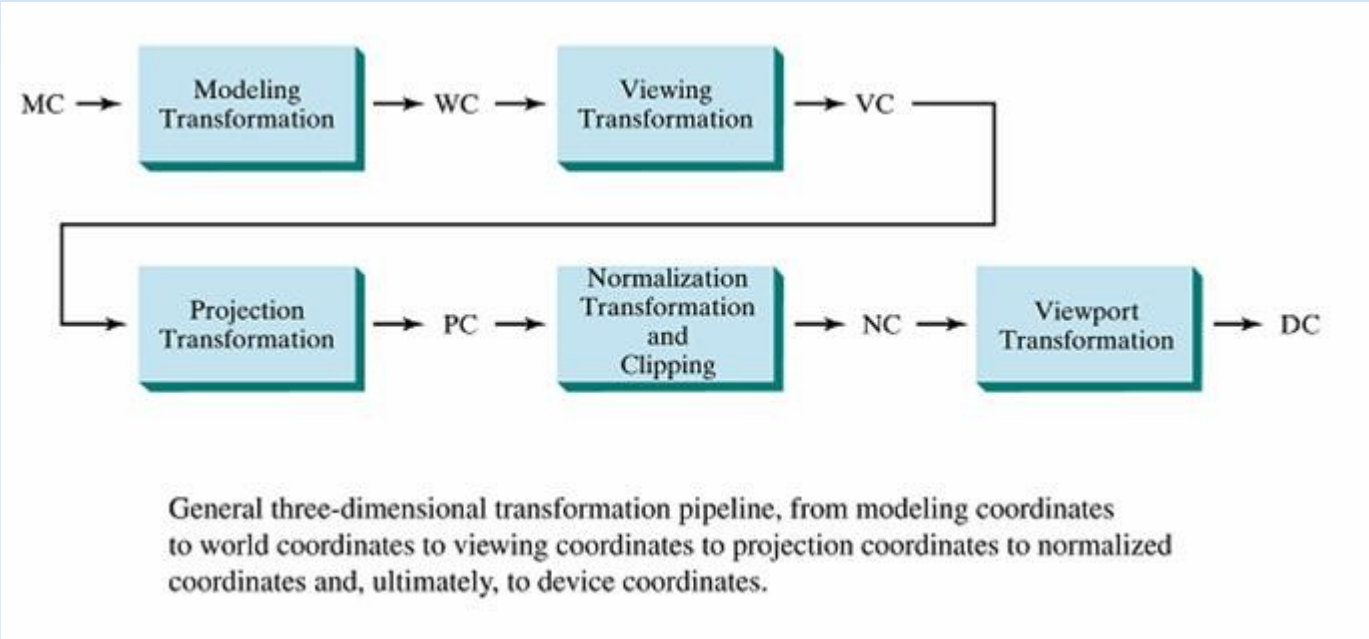
Transformaciones de Viewing 2D y 3D (Parte II)

Contenido

- Secuencia de transformaciones 3D (transformation pipeline)
- Proyecciones clásicas
  - Proyecciones en perspectiva
  - Proyecciones en paralelo
- Sistema de coordenadas de viewing
- Matriz de proyección en perspectiva simple
- Matriz de proyección en perspectiva simple
- 3D Viewing en OpenGl

3D rendering pipeline

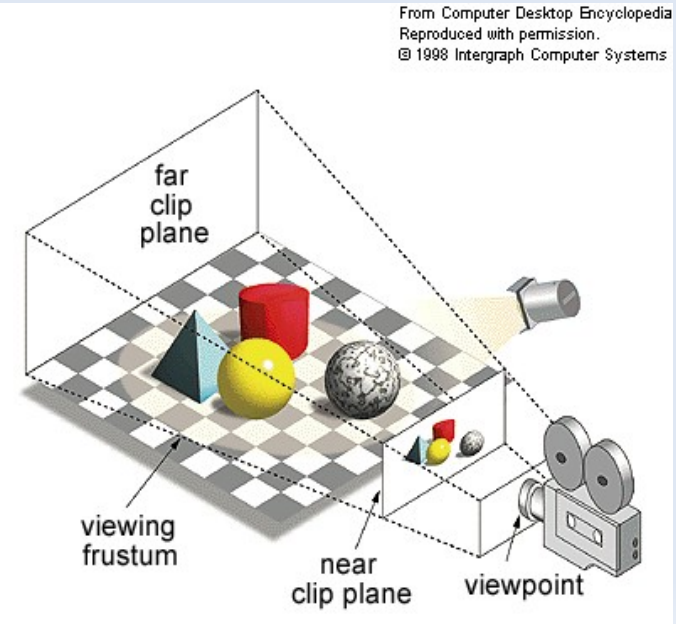
- Recordemos que, el proceso completo de rendering incluye las siguientes transformaciones:



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.

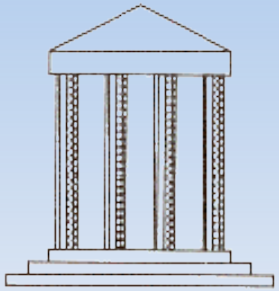
Transformación de Viewing y proyección en 3D

- Por qué es más complejo que en 2D?
  - Especificar desde donde se mira la escena (dónde posicionamos y qué orientación le damos a la “cámara”)
  - Especificar proyección (paralela o perspectiva) a utilizar: Objetos en 3D deben ser proyectados en un plano 2D
  - Especificar el “view volume”, es decir que parte de la escena se desea proyectar: sólo los objetos que están dentro de esta volumen de visión son proyectados a dos dimensiones

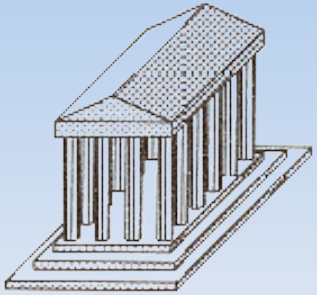


- El recorte de los objetos (algoritmos de clipping) se aplican sobre el volumen de visión

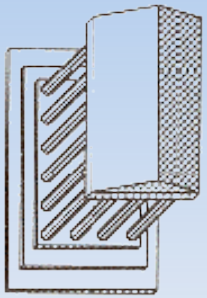
Proyecciones clásicas



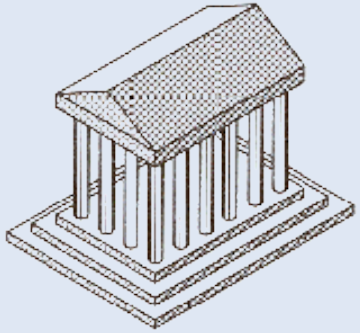
Front elevation



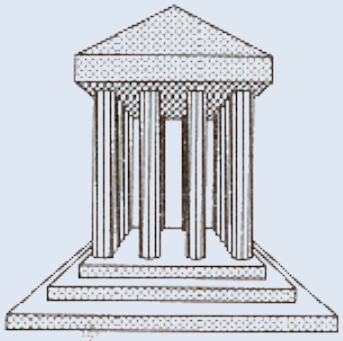
Elevation oblique



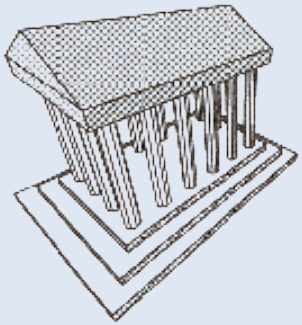
Plan oblique



Isometric

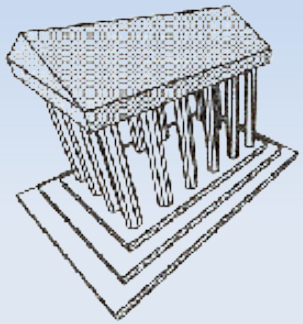


One-point perspective

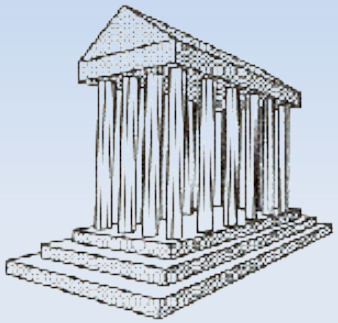


Three-point perspective

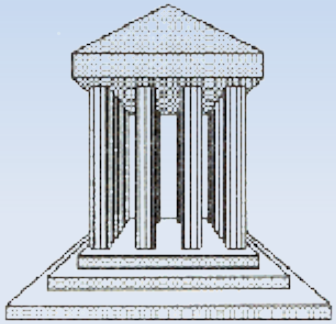
Proyecciones en perspectiva



3-Point
Perspective



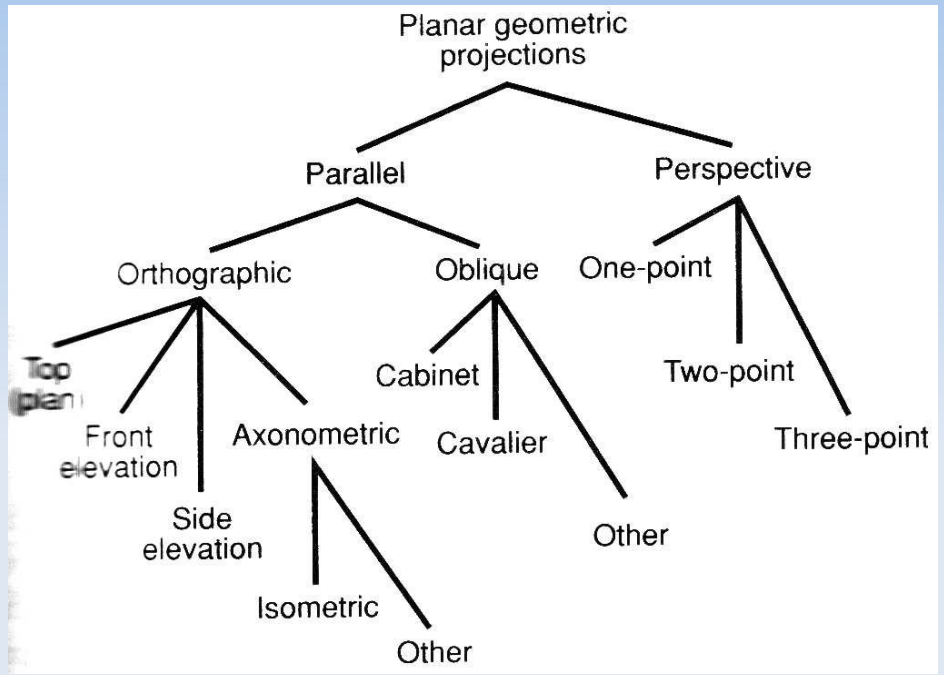
2-Point
Perspective

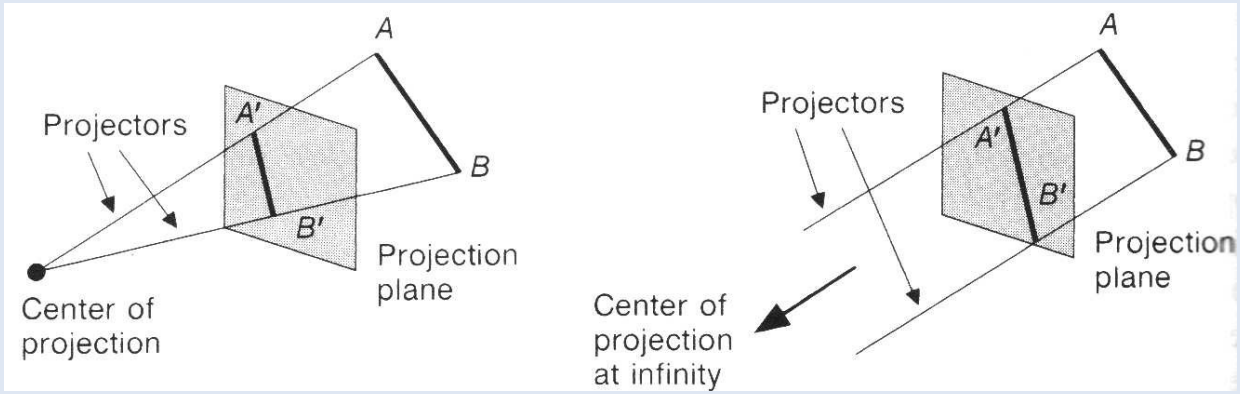


1-Point
Perspective

Clasificación de proyecciones

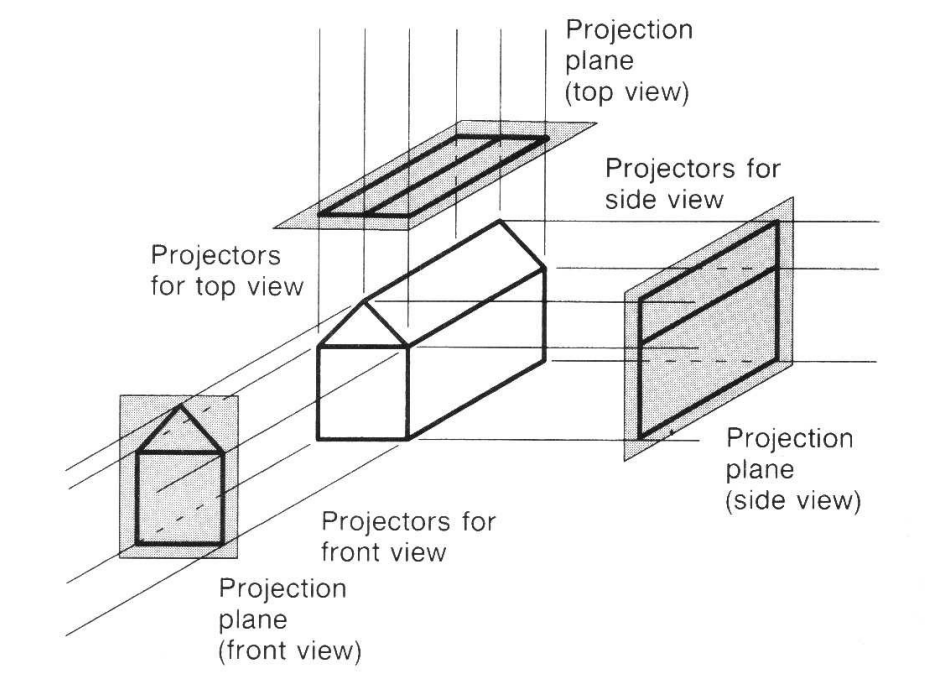
- Proyección en perspectiva:
  existe centro de proyección y proyectores no paralelos
- Proyección paralela:
  proyectores paralelos y centro de proyección en el infinito

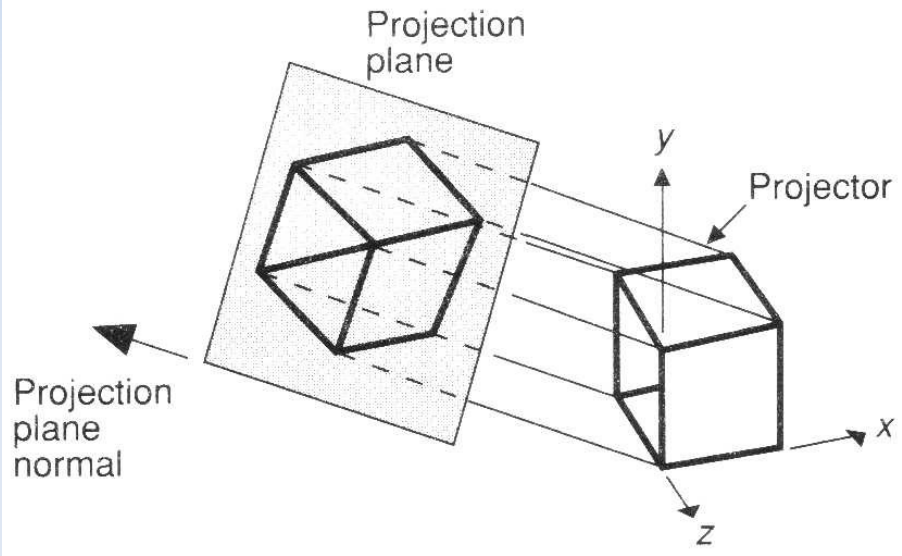




Proyecciones paralelas

Proyección ortográfica

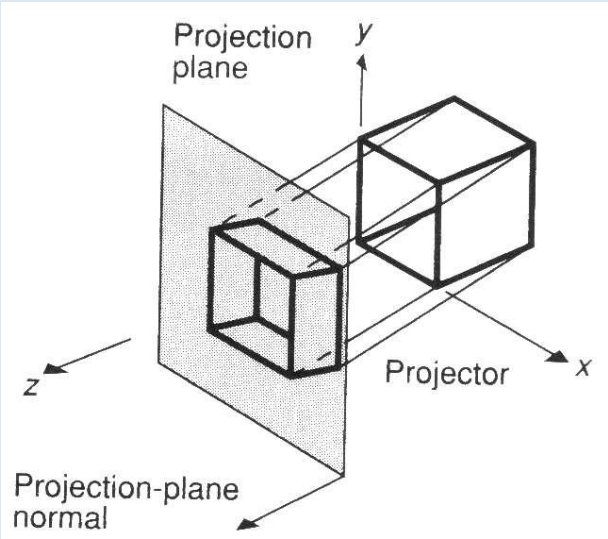




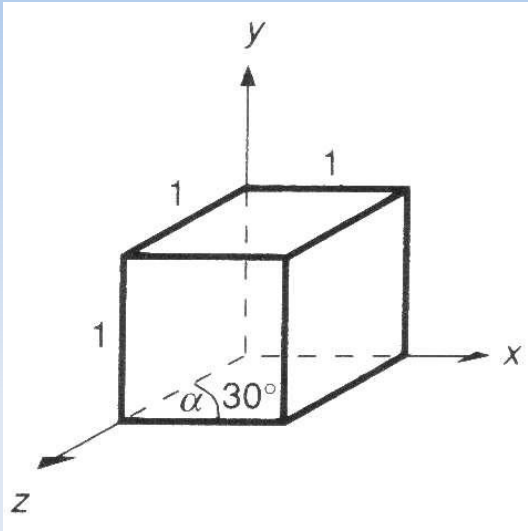
Mantiene las proporciones en los
Tres ejes principales (iso – igual y
Metric – medida)

Proyecciones paralelas oblicuas

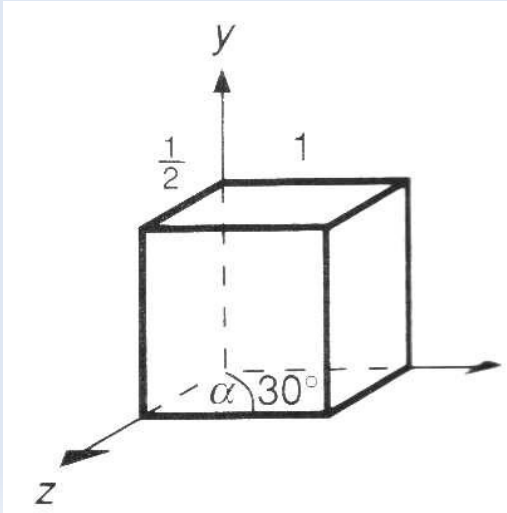
- Dirección de proyección es distinta de la dirección del vector normal al view plane



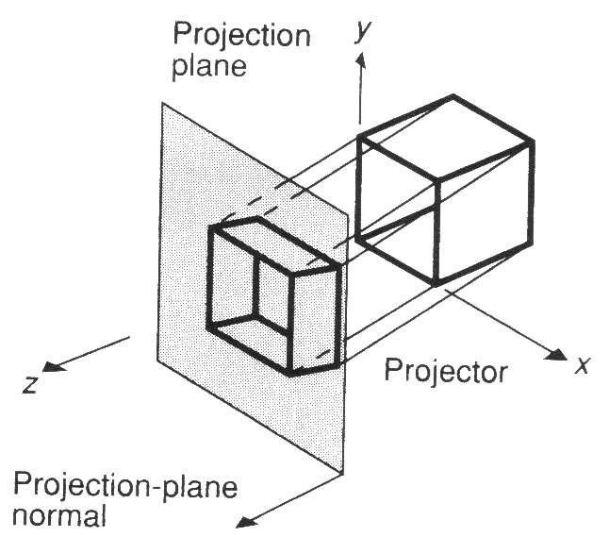
Proyección cavalier

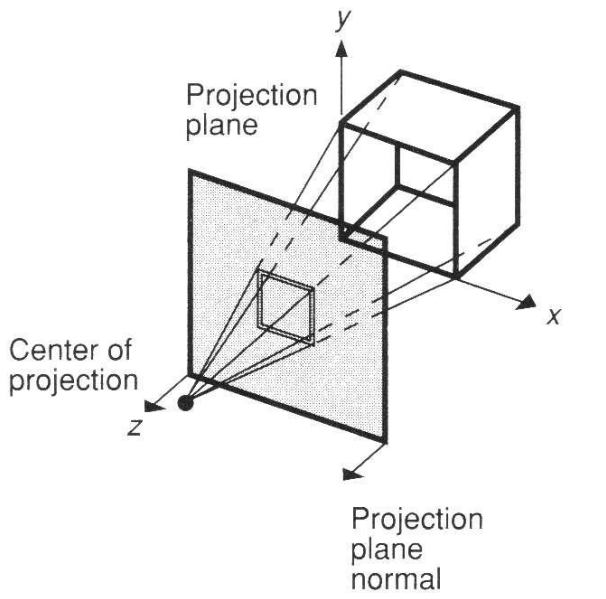


Proyección de cabinet



Proyección en paralela versus proyección en perspectiva

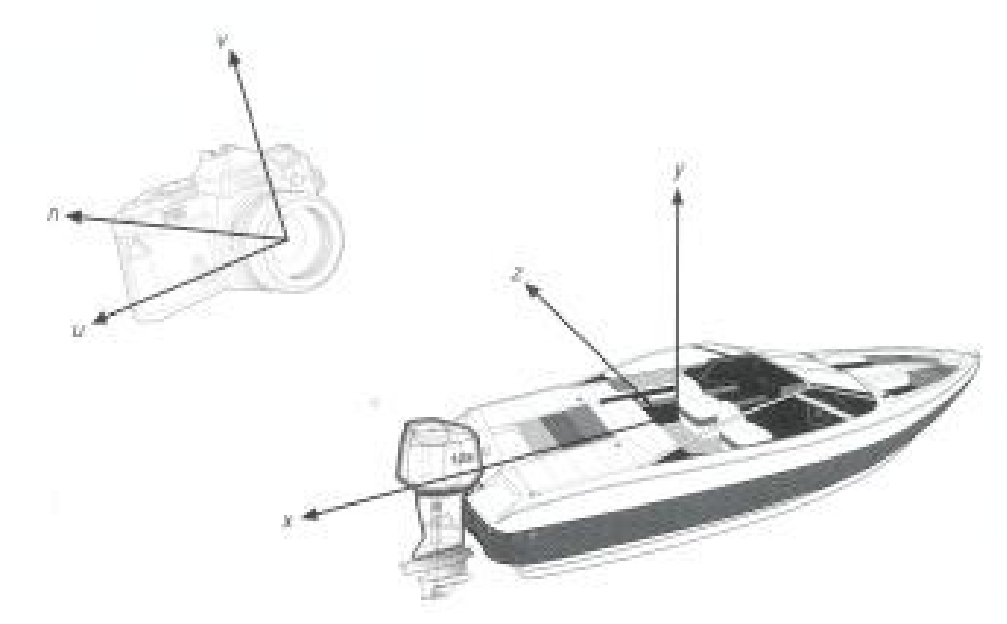


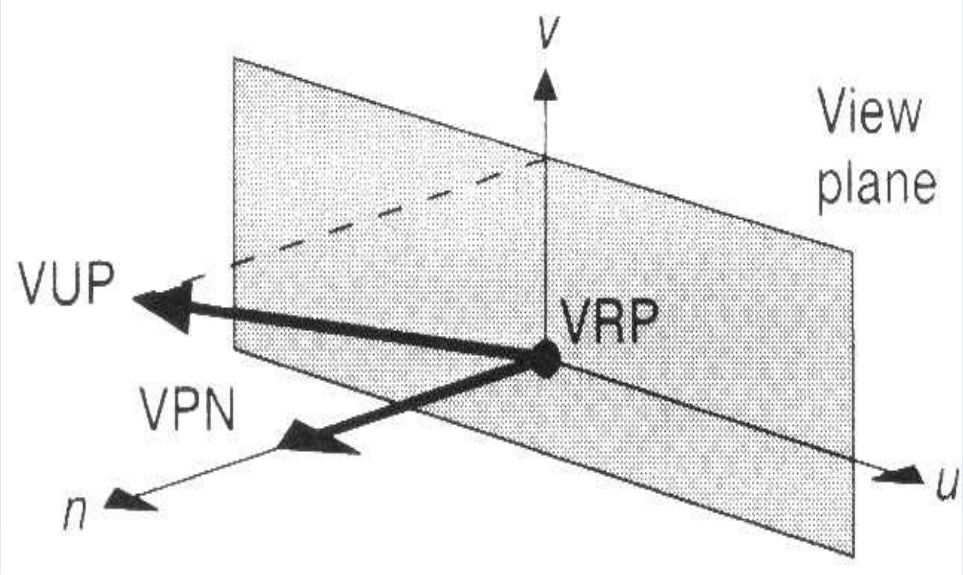


- En paralelo:
  - Visualización menos realista pero apropiada para mediciones exactas
  - Preserva paralelismo de líneas
- En perspectiva:
  - Visión más realista, tamaño varía inversamente proporcional con la distancia
  - No conserva paralelismo en las líneas

Viewing coordinates (VC)

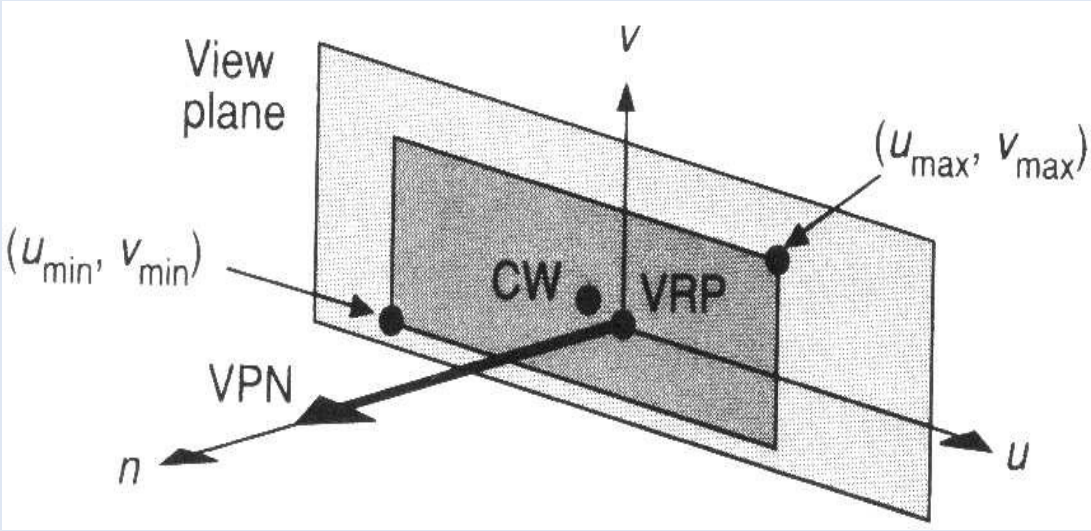
- El view reference coordinate system(VRC) se define con:
  - VRP: viewing reference point. Define el origen dell sistema de coordenadas de viewing
  - VPN(n): es el vector normal del view-plane (plano de proyección). Define dirección y sentido del eje z.
  - VUP: view up vector. Vector cuya proyección en el “view plane” (v) defin el eje y del sistema de coordenadas
  - El eje x es definido por el vector u = v x n





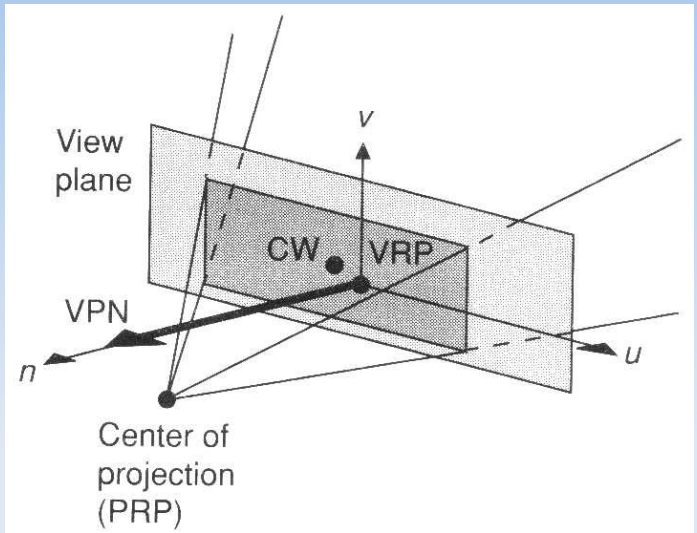
View reference coordinate system

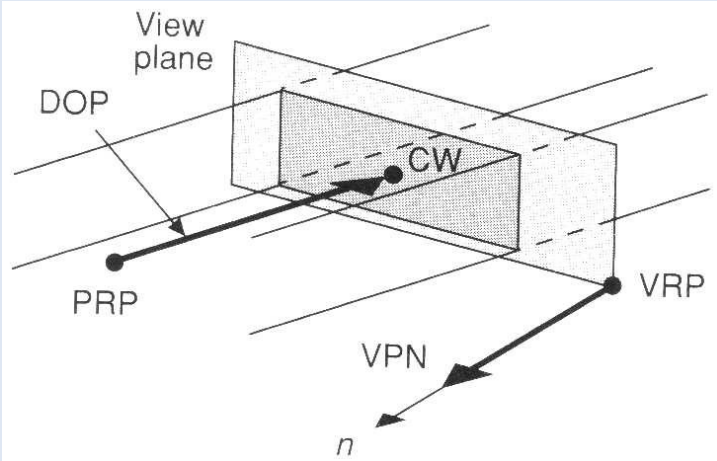
- Qué parte del view plane interesa? (Foley-Van Dam,..)
  - Ventana definida por (u_{min}, v_{min}) y (u_{max}, v_{max})
  - La ventana no necesita ser simétrica con respecto al VRP (centro en CW)



View reference coordinate system

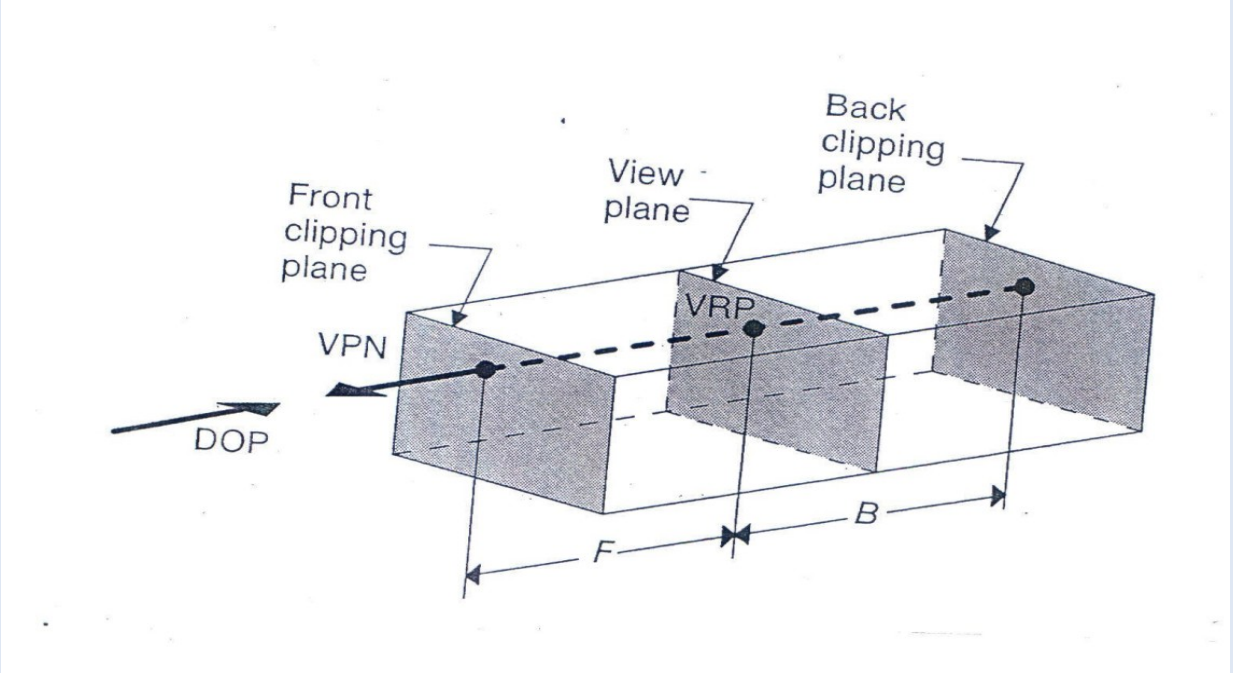
- PRP (projection reference point) y DOP (projection direction) definidos en el sistema VRC
  - Perspectiva: PRP es el centro de poryección
  - Paralela: CW-PRP define DOP





View reference coordinate system

- Proyección paralela: volumen de visión (view volume)
  - Paralelepípedo delimitado por un plano frontal y posterior
  - Define la parte de la escena a proyectar en el view plane (plano de visión)



View reference coordinate system

- Proyección perspectiva: volumen de visión (view volume)
  - Pirámide truncada por un plano frontal y posterior
  - Define la parte de la escena a proyectar en el view plane (plano de visión)

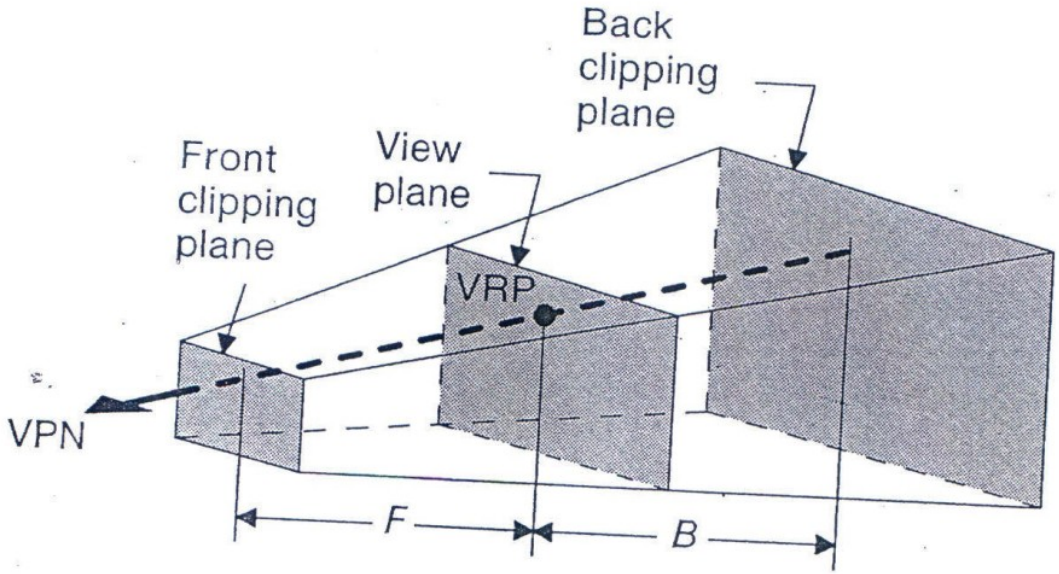


Fig. 6.21 Truncated view volume.

Deducción de matriz en perspectiva simple

- Plano de proyección (view plane) a distancia d del origen
- P: punto a proyectar
- La proyección de P es

$$P_p = (x_p, y_p, z_p)$$

- usando los triángulos semejantes, se obtien

$$\frac{x_p}{d} = \frac{x}{z} \Rightarrow x_p = \frac{x}{z/d}$$

$$\frac{y_p}{d} = \frac{y}{z} \Rightarrow y_p = \frac{y}{z/d}$$

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

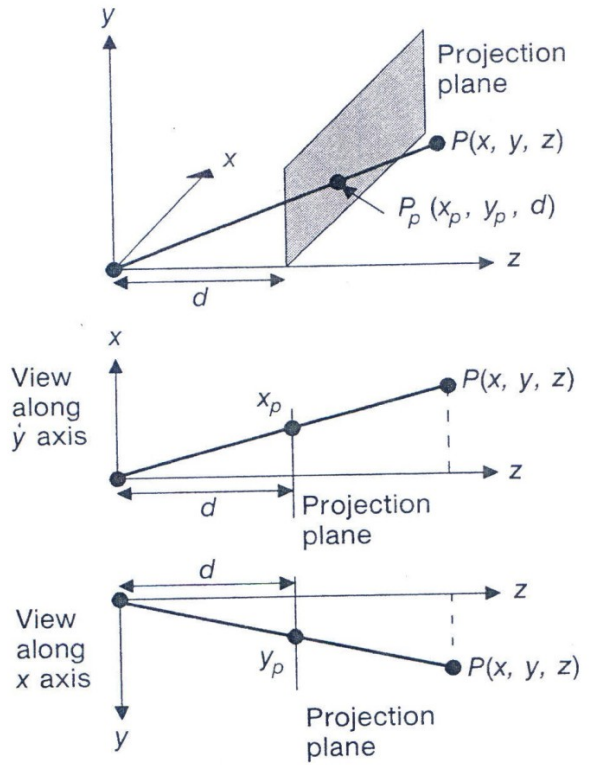


Fig. 6.42 Perspective projection.

Deducción de matriz en perspectiva simple

- Multiplicando $M_{per} \quad por \quad P=[x \quad y \quad z \quad 1]^T$

$$M_{per} \quad P= \quad [x \quad y \quad z \quad \frac{z}{d}]^T$$

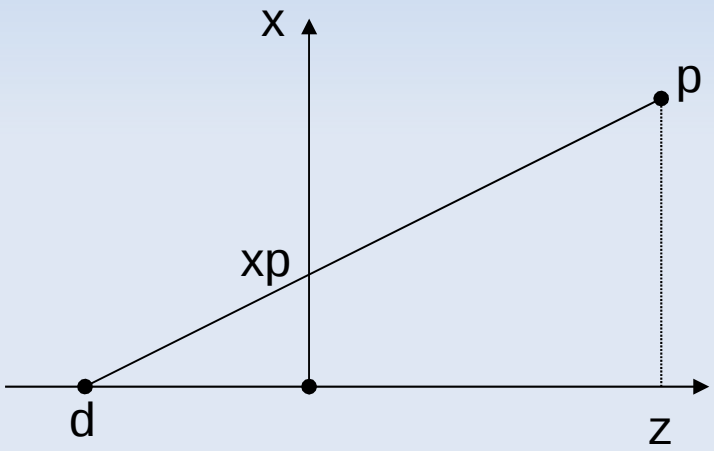
- Con $w=\frac{z}{d}$

- Luego para recuperar el punto en R^3 debemos dividir por w

$$P_p=(x_p,y_p,z_p)=(\frac{x}{z/d},\frac{y}{z/d},1)$$

Otra deducción de matriz en perspectiva

- Plano de proyección z=0, centro de proyección z= -d



$$\frac{x_p}{d} = \frac{x}{z+d} \qquad \frac{y_p}{d} = \frac{y}{z+d}$$

$$x_p = \frac{x}{z/d+1} \qquad y_p = \frac{y}{z/d+1}$$

$$M'_{per} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1/d & 1 \end{bmatrix}$$

- En este caso d puede tender a infinito... Qué significa?

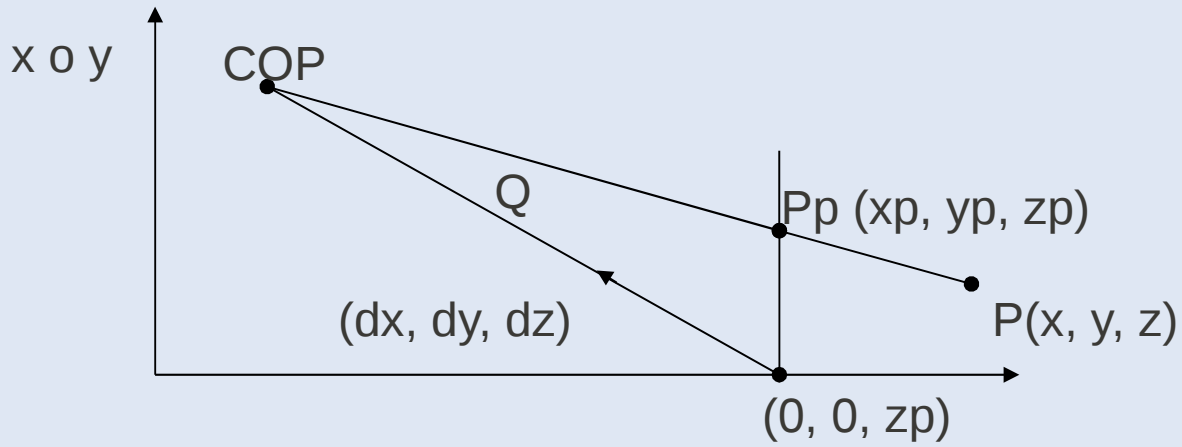
Proyección paralela ortográfica

- Plano de proyección en z= 0
  - El plano de proyección tambien en x=0 e y=0
- Dirección de proyección normal al plano de proyección
  - Si DOP = (0,0,1) y N = (0,0,1) obtenemos

$$M_{ort} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Deducción general: integra proyecciones en perspectiva y paralela

- Plano de proyección: $z = z_p$
- Centro de proyección COP a distancia Q de $(0,0,z_p)$
- Dirección desde $(0,0,z_p)$ a COP dado por vector normalizado (dx,dy,dz)
- Ecuación paramétrica línea recta de COP a P
  - PL(t) = COP + t(P-COP) 0<= t <= 1



Deducción general: integra proyecciones en perspectiva y paralela

- COP = $(0,0,z_p)+Q(d_x,d_y,d_z)$
- y reemplazandolo en la ecuación paramétrica
  - PL(t) = COP + t(P-COP) 0<= t <= 1
- Obtenemos los puntos de la recta P' = (x',y',z') son
  - $x'=Qd_x+(x-Qd_x)t$
  - $y'=Qd_y+(y-Qd_y)t$
  - $z'=(z_p+Qd_z)+(z-(z_p+Qd_z))t$
- Pp, el punto de intersección entre PL(t) y el plano de proyección $z=z_p$ se obtiene haciendo $z'=z_p$
- $$t=\frac{z_p-(z_p+Qdz)}{z-(z_p+Qdz)}$$

Deducción general: integra proyecciones en perspectiva y paralela

- Para obtener el mismo denominador D en z_p

  se multiplica

$$z_p = z_p \frac{D}{D}$$

$$x_p = \frac{x - z\,\frac{d_x}{d_z} + z_p\,\frac{d_x}{d_z}}{D}$$

$$y_p = \frac{y - z\,\frac{d_y}{d_z} + z_p\,\frac{dy}{d_z}}{D}$$

- Propuesto: obtener los valores de xp, yp, zp

$$z_p = \frac{-z\,\frac{z_p}{Qd_z} + \frac{z_p^{\,2} + z_p\,Qd_z}{Qd_z}}{D}$$

$$\text{donde } D = \frac{z_p - z}{Qd_z} + 1$$

Deducción general: integra proyecciones en perspectiva y paralela

- La matriz general es:

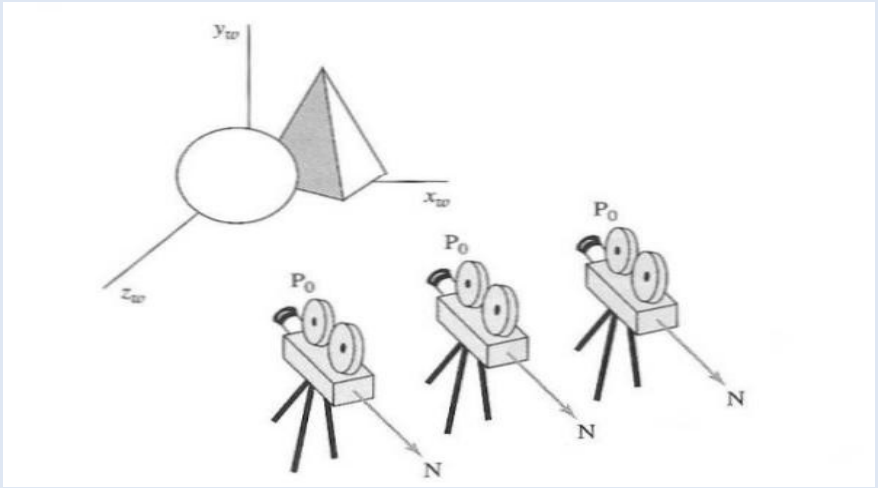
$$M_{general} = \begin{bmatrix} 1 & 0 & -\frac{d_x}{d_z} & z_p \frac{d_x}{d_z} \\ 0 & 1 & -\frac{d_y}{d_z} & z_p \frac{d_y}{d_z} \\ 0 & 0 & -\frac{z_p}{Qd_z} & \left(\frac{z_p^2}{Qd_z} + z_p \right) \\ 0 & 0 & -\frac{1}{Qd_z} & \left(\frac{z_p}{Qd_z} + 1 \right) \end{bmatrix}$$

- Para obtener las matrices anteriores

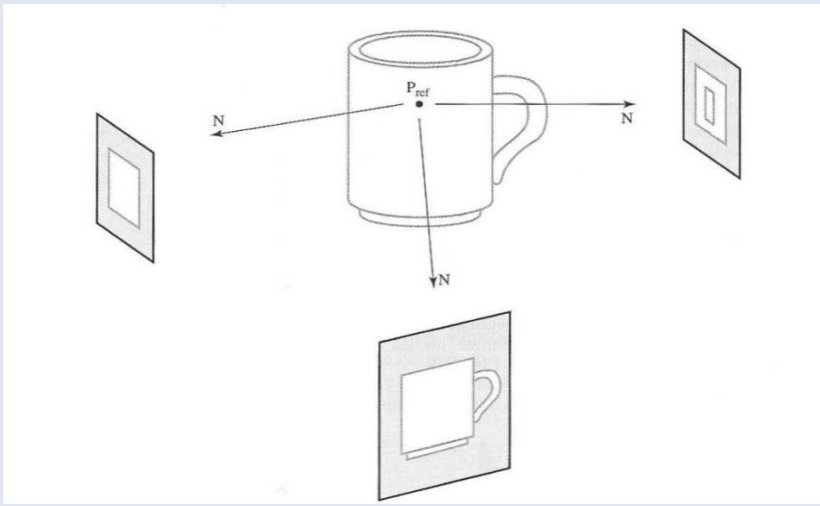
	z_p	Q	$(d_x,$	$d_y,$	$d_z)$
M_{ort}	0	∞	(0,	0,	-1)
M_{per}	d	d	(0,	0,	-1)
M'_{per}	0	d	(0,	0,	-1)

OpenGL 3D viewing-transformation

- Los parámetros de viewing se especifican con:
  - gluLookAt(x0,y0,z0,zref,yref,zref,vx,vy,vz);
  - VRP = P0(x0,y0,z0) origen del sistema de coordenadas en WC
  - Pref = punto de referencia (normalmente se elige como un punto que uno quiere observar de la escena)
  - Vup = (vx,vy,vz) y N = P0-Pref
  - Valores iniciales: P0=(0,0,0), Pref = (0,0,-1) y Vup =(0,1,0)



P0 cambia , N fijo



Pref fijo, N cambia

OpenGl proyecciones

- Para especificar una proyección se debe pasar al modo “proyección”
  - glMatrixMmode(GL_PROJECTION)
- Proyección ortogonal
  - glOrtho(xwmin,xwmax,ywmin,ywmax,dnear,dfar)
  - (xwmin,ywmin) y (xwmax,ywmax) es la ventana en el view plane y se ubica en el plano frontal de clipping (dnear)
  - dnear y dfar la posición de los plano frontal y posterior (define volumen de visión)
  - glOrtho(-1.0,1.0,-1.0,1.0,-1.0,1.0) (valores iniciales)

Proyección paralela

- Los parámetros de la proyección paralela

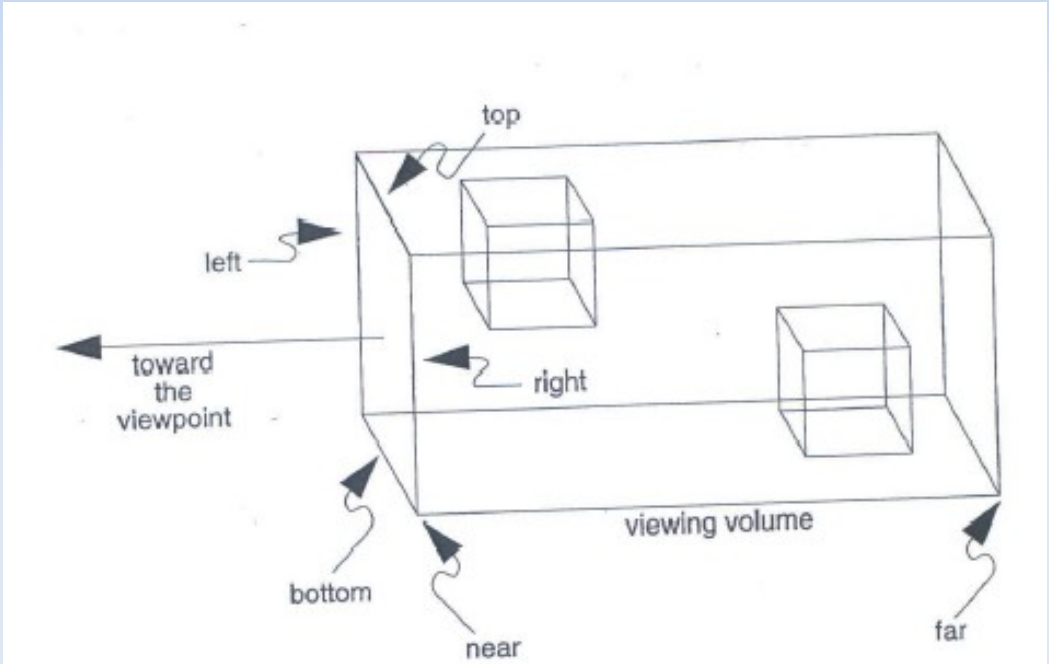


Figure 3-15 Orthographic Viewing Volume

Proyección en Perspectiva

- Una proyección en perspectiva general se especifica con
  - gluFrustum(xwmin,xwmax,ywmin,ywmax,dnear,dfar)

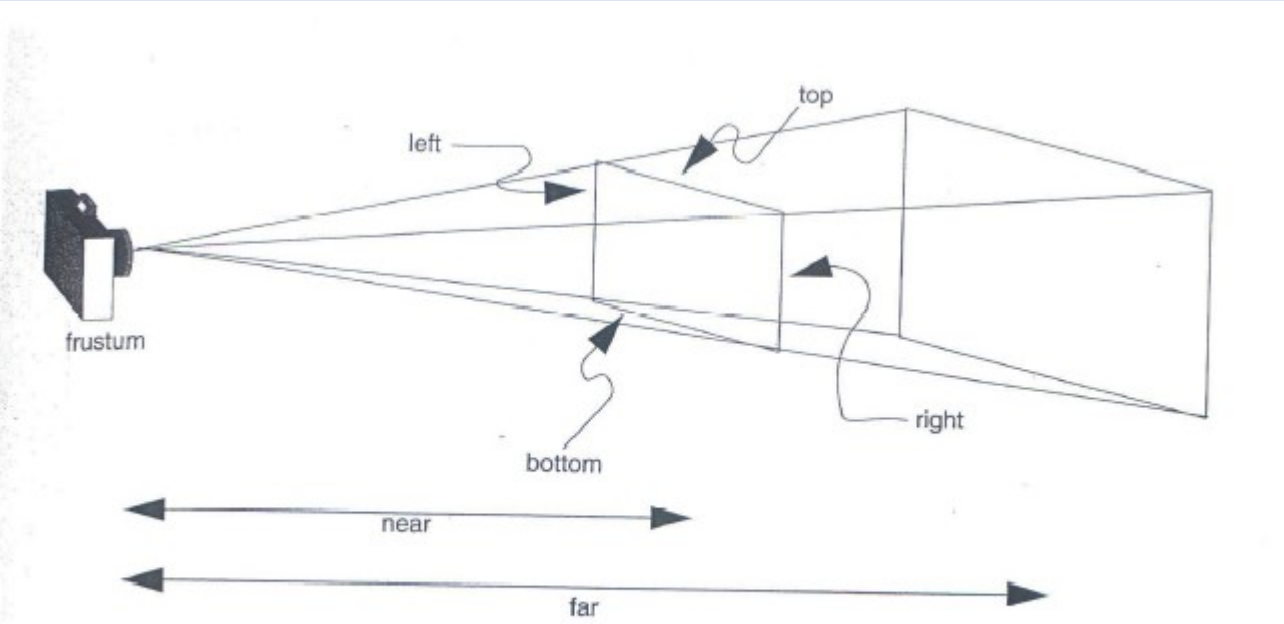
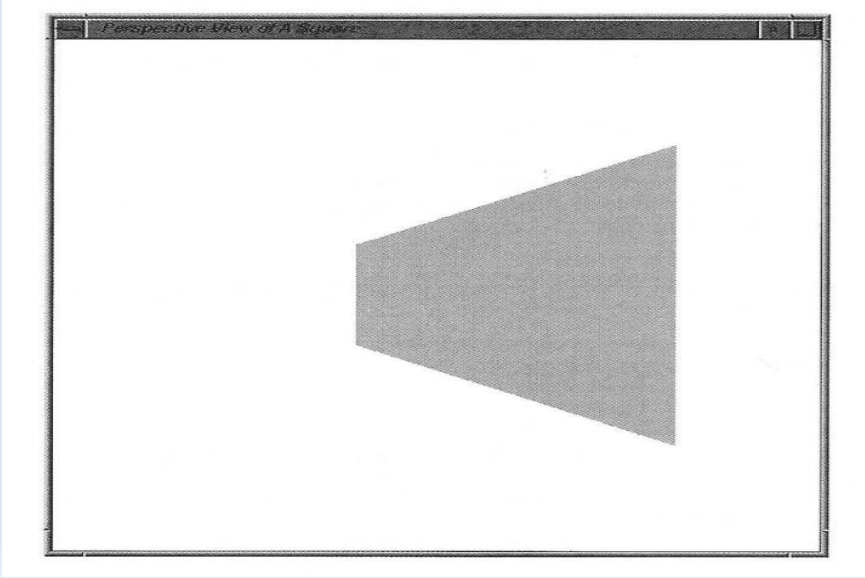


Figure 3-13 Perspective Viewing Volume Specified by glFrustum()

Proyección en Perspectiva

- Frustum es el volumen de vista
- dnear y dfar: distancia desde el punto de vista al plano frontal y posterior, respectivamente (internamente de consideran -dnear y -dfar)
- La coordenadas (left,bottom) y (rigth,top) definen la ventana en el view-plane. El view-plane se posiciona en el plano frontal (dnear)
- No define necesariamente un volumen de vista simétrico

- Ejemplo: Programa que visualiza un cuadrado en perpectiva



Programa en OpenGL

```
#include < GL/glut.h>
GLint winWidth = 600, winHeight = 600; // Tamaño inicial ventana de visualización.
GLfloat x0 = 100.0, y0 = 50.0, z0 = 50.0; // Origen coordenadas de visualización.
GLfloat xref = 50.0, yref = 50.0, zref = 0.0; // Punto observado.
GLfloat Vx = 0.0, Vy = 1.0, Vz = 0.0; // Vector vertical.
```

```
/* Establecer límites de coordenadas para ventana de recorte: */
GLfloat xwMin = -40.0, ywMin = -60.0, xwMax = 40.0, ywMax = 60.0;

/* Establecer posición de los planos de recorte próximo y lejano: */
GLfloat dnear = 25.0, dfar = 125.0;

void init (void)
{
   glClearColor (1.0, 1.0, 1.0, 0.0);
   glMatrixMode (GL_MODELVIEW);
   gluLookAt (x0, y0, z0, xref, yref, zref, Vx, Vy, Vz);
   glMatrixMode (GL_PROJECTION);
   glFrustum (xwMin, xwMax, ywMin, ywMax, dnear, dfar);
}
```

Programa en OpenGL

```
void displayFcn (void)
{
    glClear (GL_COLOR_BUFFER_BIT);

    /* Establecer parámetros para un área de relleno cuadrada. */
    glColor3f (0.0, 1.0, 0.0); // Seleccionar color de relleno verde.
    glPolygonMode (GL_FRONT, GL_FILL);
    glPolygonMode (GL_BACK, GL_LINE); // Cara posterior alámbrica.
    glBegin (GL_QUADS);
        glVertex3f (0.0, 0.0, 0.0);
        glVertex3f (100.0, 0.0, 0.0);
        glVertex3f (100.0, 100.0, 0.0);
        glVertex3f (0.0, 100.0, 0.0);
    glEnd ( );
    glFlush ( );
}
```

```
void reshapeFcn (GLint newWidth, GLint newHeight)
{
    glViewport (0, 0, newWidth, newHeight);

    winWidth = newWidth;
    winHeight = newHeight;
}
void main (int argc, char** argv)
{
    glutInit (&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
    glutInitWindowPosition (50, 50);
    glutInitWindowSize (winWidth, winHeight);
    glutCreateWindow ("Perspective View of A Square");

    init ( );
    glutDisplayFunc (displayFcn);
    glutReshapeFunc (reshapeFcn);
    glutMainLoop ( );
}
```

Proyección en Perspectiva

- Una proyección en perspectiva simétrica se especifica con:
- gluPerspective(theta,aspect,dnear,dfar)

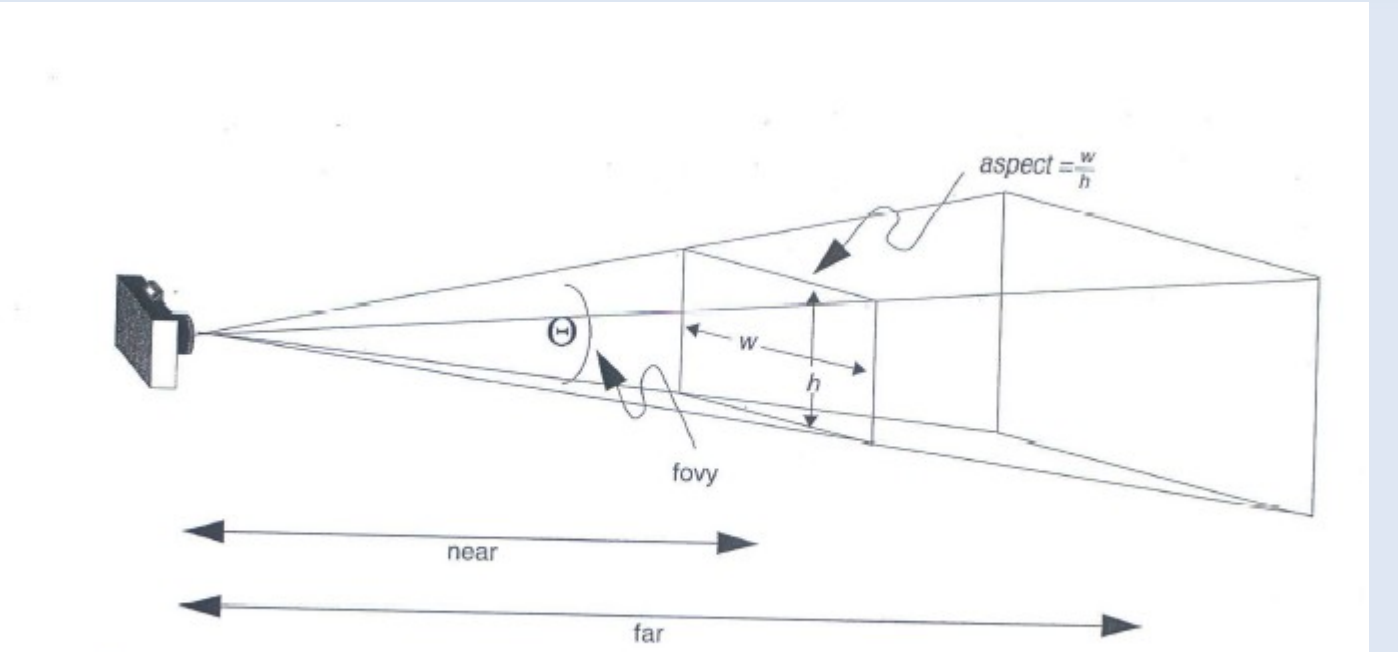


Figure 3-14 Perspective Viewing Volume Specified by gluPerspective()

Proyección en Perspectiva

- Volumen de vista simétrico. Parámetros:
  - Fovy ángulo del campo de vista, entre 0 y 180
  - Aspect: razón de aspecto de la ventana en el view plane (w/h)
  - Near y far: distancia desde el origen del sistema de coordenadas (view point) al plano frontal y posterior, respectivamente

Proyección en Perspectiva

Even after you've aimed the camera in the correct direction and you can see your objects, they might appear too small or too large. If you're using **gluPerspective()**, you might need to alter the angle defining the field of view by changing the value of the first parameter for this command. You can use trigonometry to calculate the desired field of view given the size of the object and its distance from the viewpoint: The tangent of half the desired angle is half the size of the object divided by the distance to the object (see Figure 3-19). Thus, you can use an arctangent routine to compute half the desired angle. Example 3-3 assumes such a routine, **atan2()**, which calculates the arctangent given the length of the opposite and adjacent sides of a right triangle. This result then needs to be converted from radians to degrees.

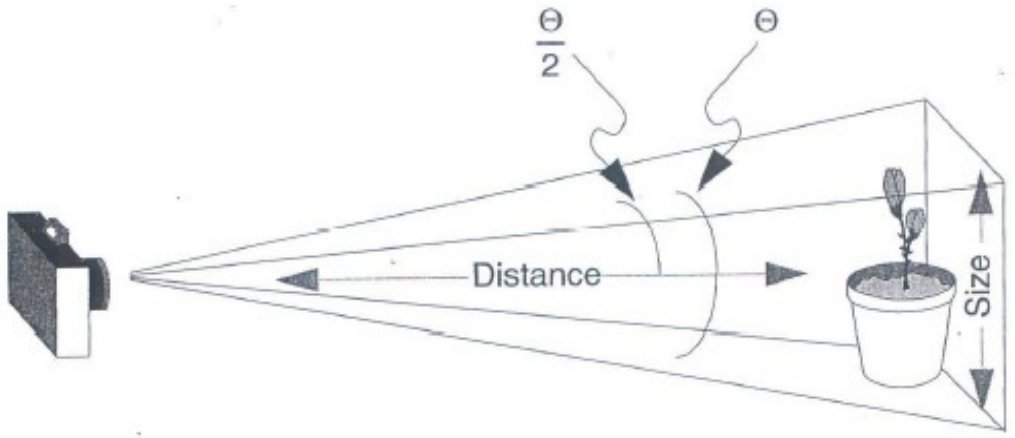


Figure 3-19 Using Trigonometry to Calculate the Field of View

Proyección en Perspectiva

Example 3-3 Calculating Field of View

```
#define PI 3.1415926535

double calculateAngle(double size, double distance)
{
    double radtheta, degtheta;

    radtheta = 2.0 * atan2 (size/2.0, distance);
    degtheta = (180.0 * radtheta) / PI;
    return (degtheta);
}
```

Of course, typically you don't know the exact size of an object, and the distance can only be determined between the viewpoint and a single point in your scene. To obtain a fairly good approximate value, find the bounding box for your scene by determining the maximum and minimum x, y, and z coordinates of all the objects in your scene. Then calculate the radius of a bounding sphere for that box, and use the center of the sphere to determine the distance and the radius to determine the size.