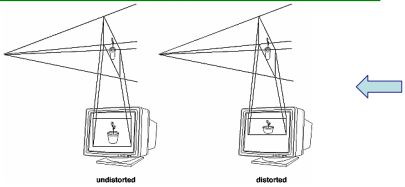
# Projeções

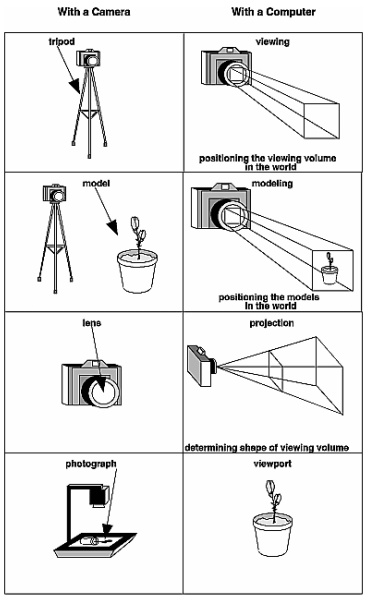
Rogers & Adams – Capítulo 3-11 a 3-17

Apostila – Capítulo 5

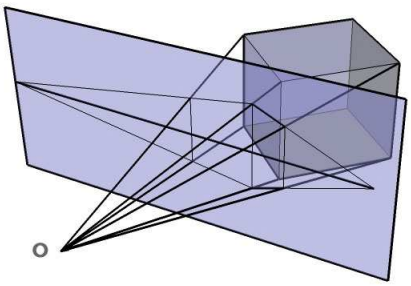
# Transformações Geométricas

- Posicionar os blocos constituintes de uma cena
  - Alterar as coordenadas dos pontos
- Projetar a cena sobre o plano de imagem
  - Alterar as coordenadas dos pontos
- Enquadrar a cena na janela de exibição
  - Alterar as coordenadas dos pontos

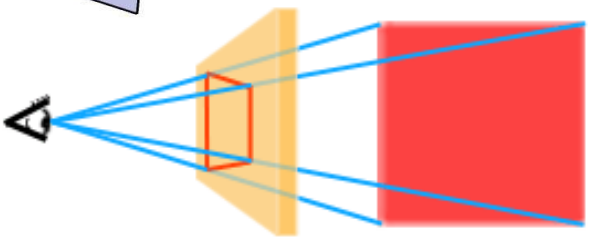


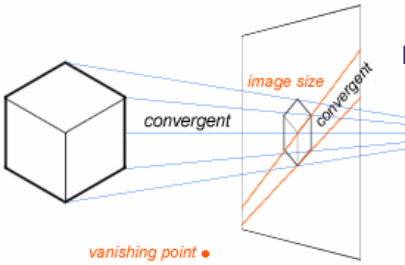
# Transformações Projetivas



Projetar modelos geométricos 3D numa imagem 2D, exibível em dispositivos de saída 2D, a fim de produzir imagens sintéticas.



# Uma Visão Clássica

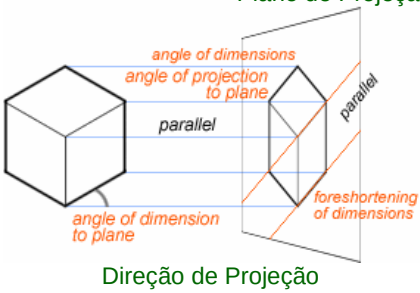



Projeções Perspectivas

Raios projetores convergentes

Centro de Projeção

Plano de Projeção




Projeções Paralelas

x

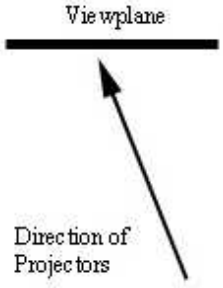
Raios projetores paralelos

Preserva paralelismo

Direção de Projeção

# Projeções Paralelas

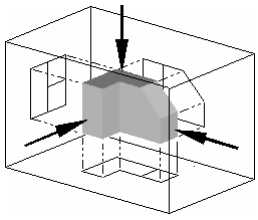
**Ortográficas/Axonométricas**



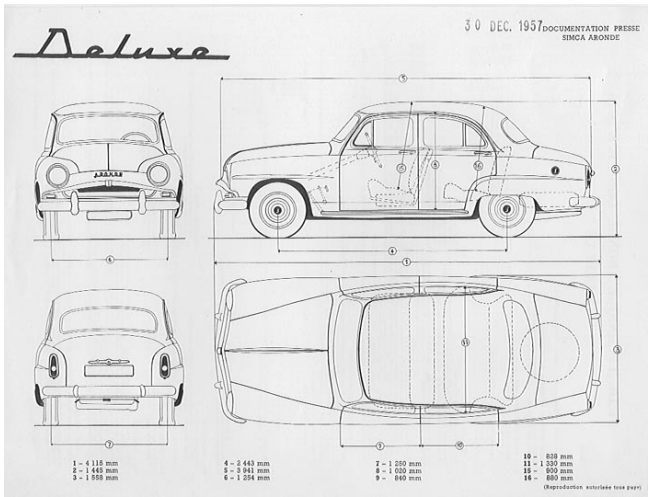

**Oblíquas**

# Projeções Ortográficas

Desenhos técnicos: preserva a relação das medidas

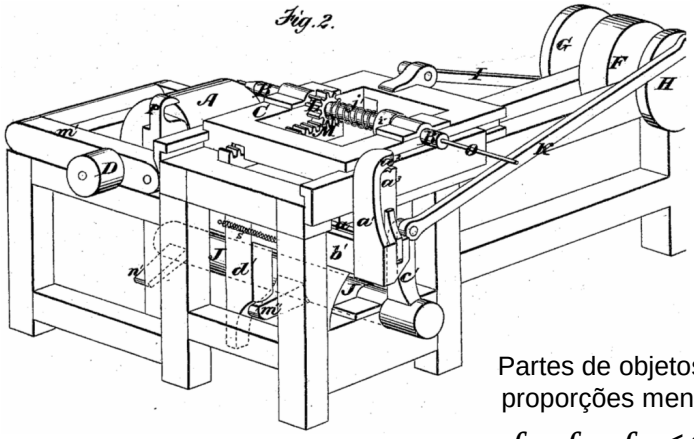


$$f_x = 1$$
$$f_y = 1$$
$$f_z = 1$$



# Projeções Axonométricas

Escorços: provêem melhor percepção de profundidade

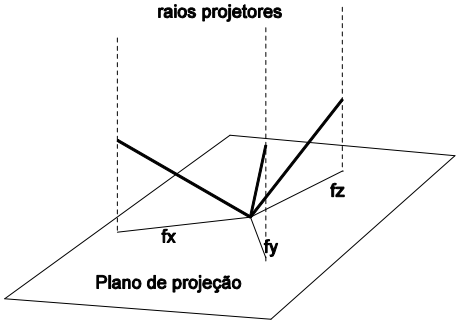



Partes de objetos em proporções menores

$$f_x, f_y, f_z \leq 1.0$$

# Projeções Axonométricas

**fator de redução < 1.0**



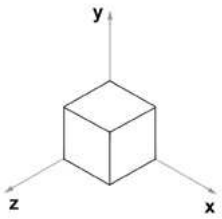

$$T \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} x_x & y_x & z_x \\ x_x & y_x & z_x \\ 0 & 0 & 0 \end{pmatrix}$$

$$f_x = \sqrt{x_x^2 + y_x^2}$$

$$f_y = \sqrt{x_y^2 + y_y^2}$$

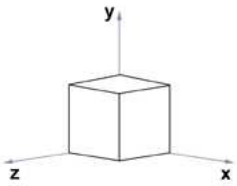
$$f_z = \sqrt{x_z^2 + y_z^2}$$

# Projeções Axonométricas



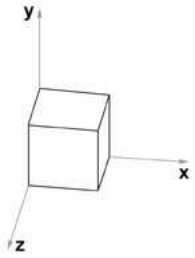

Isométricas

$$f_x = f_y = f_z$$




Dimétricas

$$f_x = f_y$$
$$f_x = f_z$$
$$f_z = f_y$$

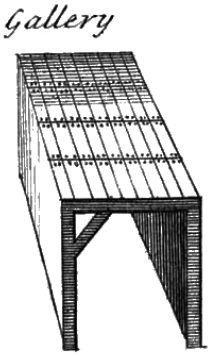


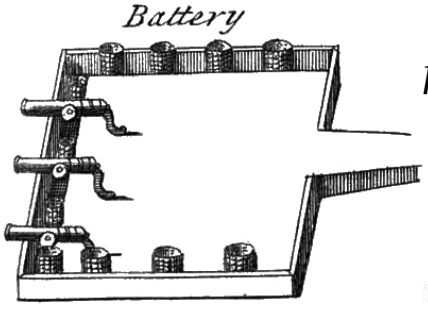

Trimétricas

$$f_x \neq f_y \neq f_z$$

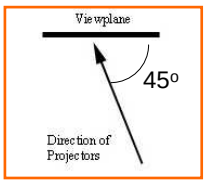
# Projeções Oblíquas Cavalier

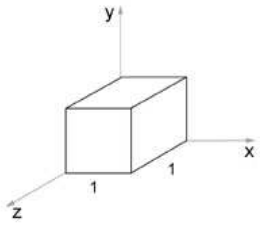
"Vista Aérea": ângulo 45º



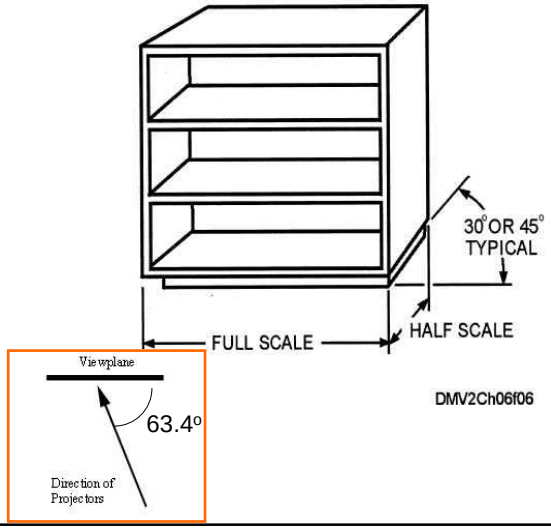

$$f_z = 1$$



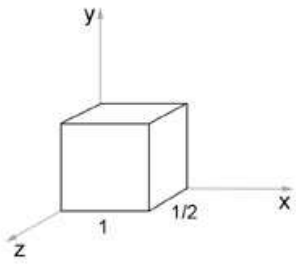



# Projeções Oblíquas Cabinet

Vista “oblíqua” de estantes: ângulo 63.4º

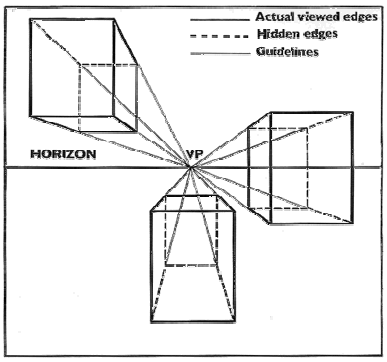



$$f_z = \frac{1}{2}$$



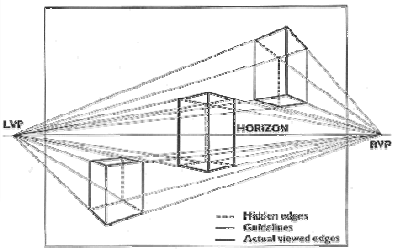

# Projeções Perspectivas

Um ponto de fuga



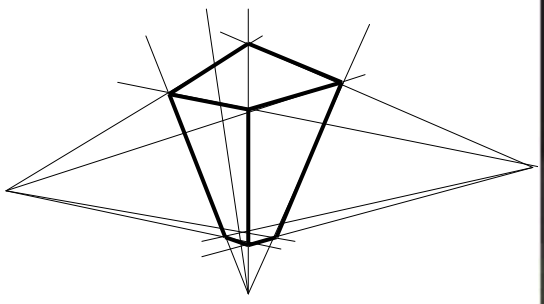
# Projeções Perspectivas

Dois pontos de fuga

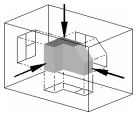



# Projeções Perspectivas

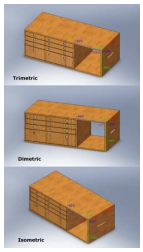
Três pontos de fuga

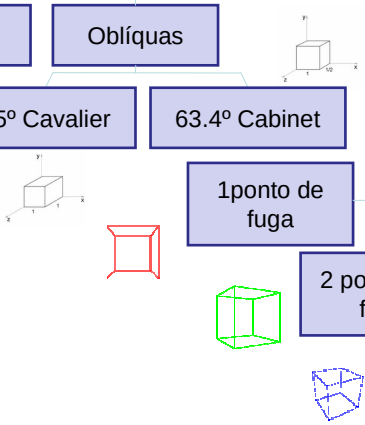


# Uma Visão Clássica

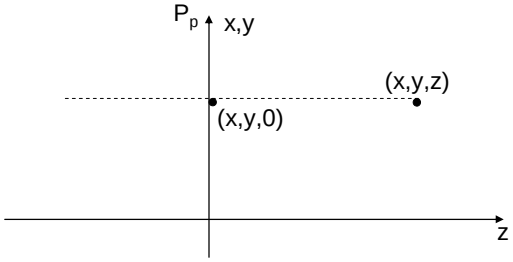


- Projeções Planas
  - Paralelas
    - Ortográficas
    - Axonométricas
      - Trimétricas
      - Dimétricas
      - Isométricas
    - Oblíquas
      - 45º Cavalier
      - 63.4º Cabinet
  - Perspectivas
    - 1ponto de fuga
    - 2 pontos de fuga
    - 3 pontos de fuga



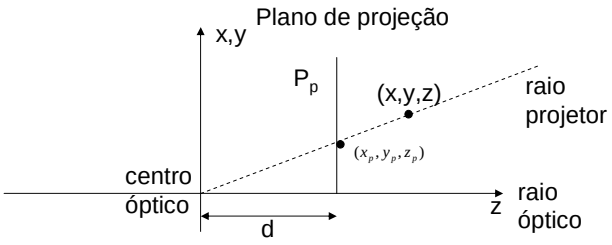


# “Algebrização”




$$\begin{bmatrix} x \\ y \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

# “Algebrização”




$$x_p = \frac{xd}{z}$$

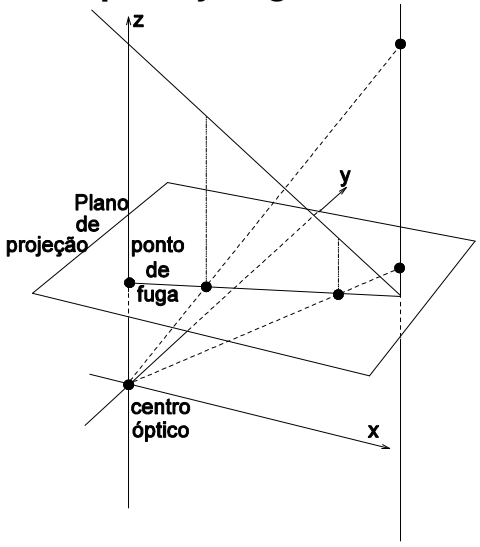
$$y_p = \frac{yd}{z}$$

$$z_p = \frac{zd}{z} = d$$

$$\begin{bmatrix} x \\ y \\ z \\ \frac{z}{d} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{d} & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

# Projeções Perspectivas

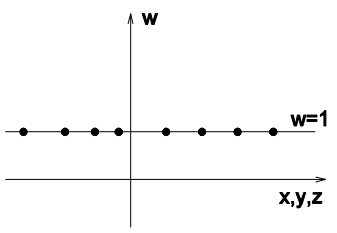
## Interpretação geométrica

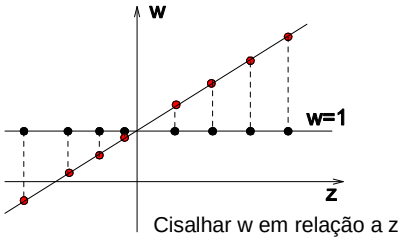



www.math.utah.edu/~treiberg/Perspect/Perspect.htm

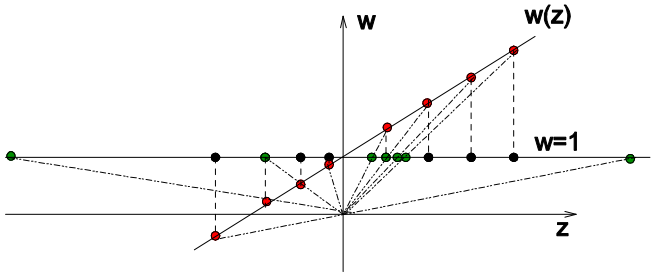
# Projeções Perspectivas

## Interpretação geométrica



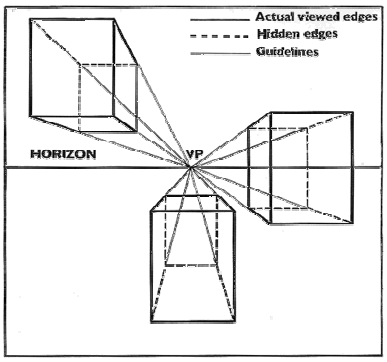

Cisalhar w em relação a z




# Projeções Perspectivas

Um ponto de fuga

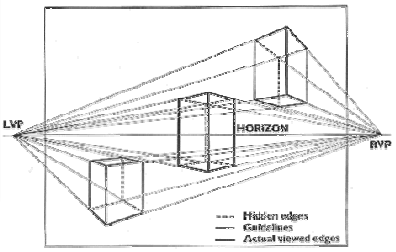
Cisalhar w em relação a uma coordenada



# Projeções Perspectivas

Dois pontos de fuga

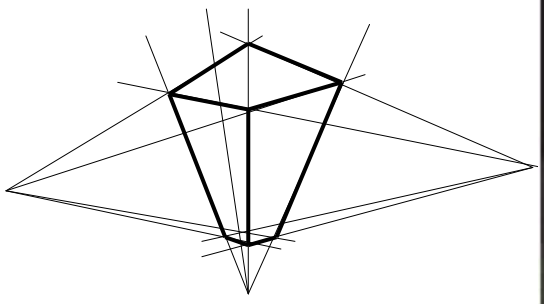
Cisalhar w em relação
a duas coordenadas



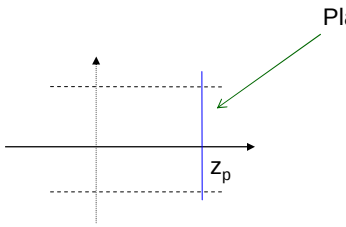

# Projeções Perspectivas

Três pontos de fuga

Cisalhar w em relação
a três coordenadas

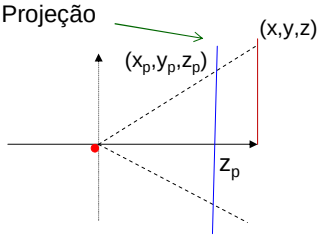


# Projeções
## Casos Triviais




Paralela

Basta substituir a coordenada z de cada ponto por $z_p$

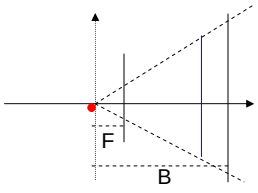


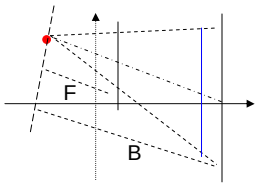

Perspectiva

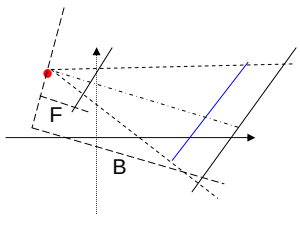
Coordenadas x, $x_p$, y e $y_p$ coincidem com as "alturas" dos triângulos e as coordenadas z com as "bases" dos triângulos. Problema se reduz a obter relação entre estas coordenadas pela semelhança de triângulos

# Projeções
## Diversidade de Casos

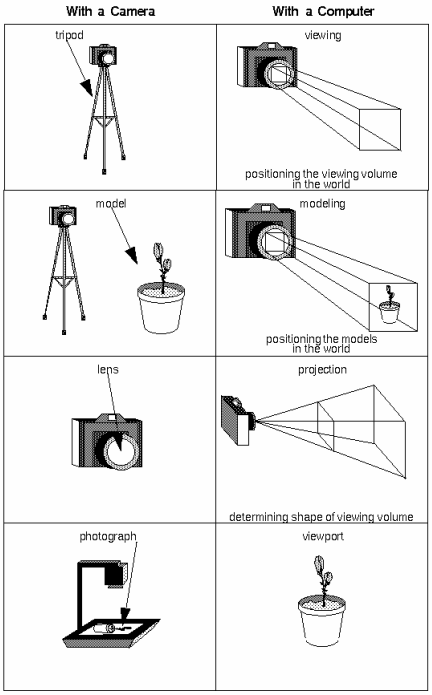
- Plano de projeção tem o vetor normal na direção do eixo z e o centro de projeção sobre o eixo z.
- Plano de projeção tem o vetor normal na direção do eixo z e o centro de projeção arbitrariamente posicionado.
- Tanto o plano quanto o centro são arbitrariamente posicionados no espaço.



## Paradigma: dividir para conquistar

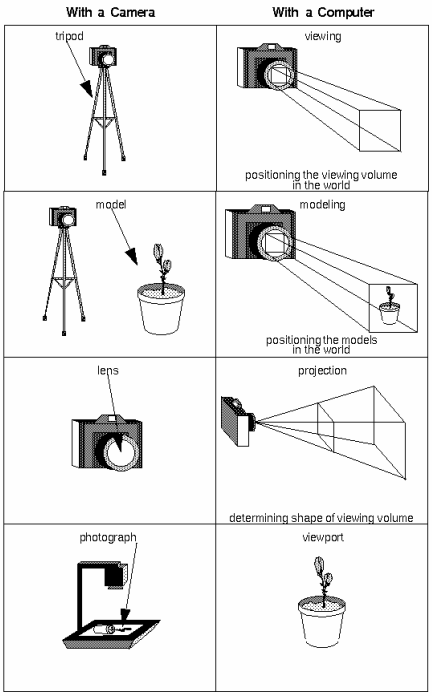



## Distintos Espaços

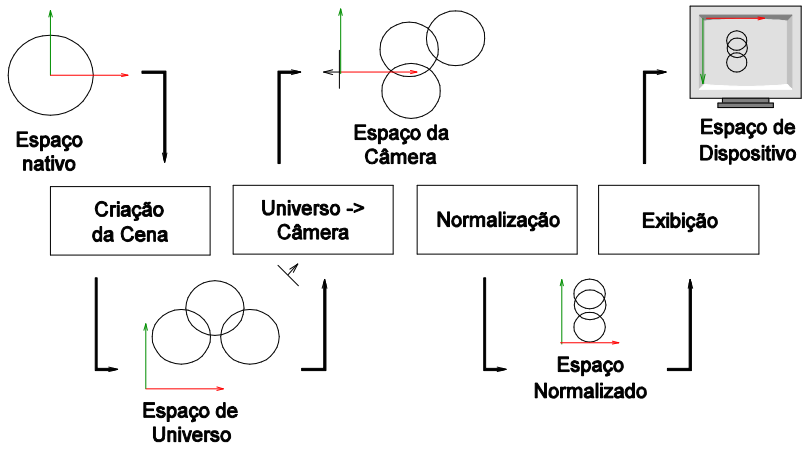
Referencial da Câmera
VRC

Referencial Normalizado
NDC

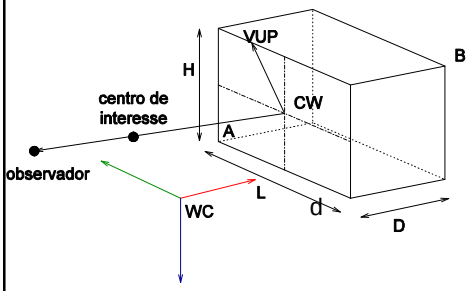
Referencial da Janela
DC



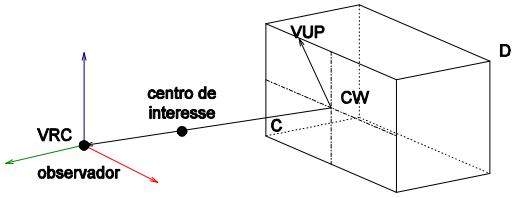
# Fluxo de Projeção




# Modelo de Câmera 1

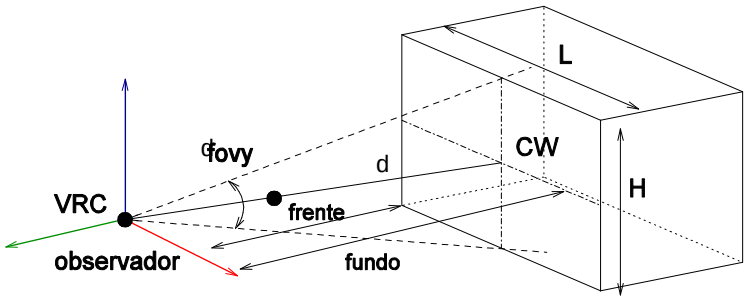



Centrado no observador



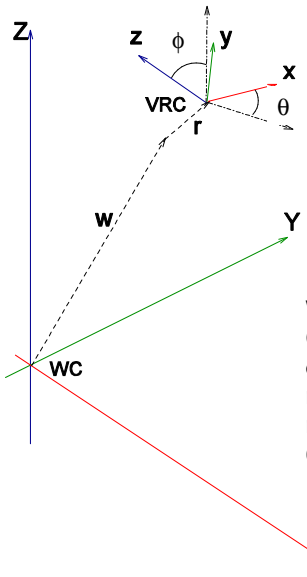

Eixo óptico coincidente com o vetor normal do plano de projeção

# Modelo de Câmera 1




d = H/(2 tg(fovy/2))

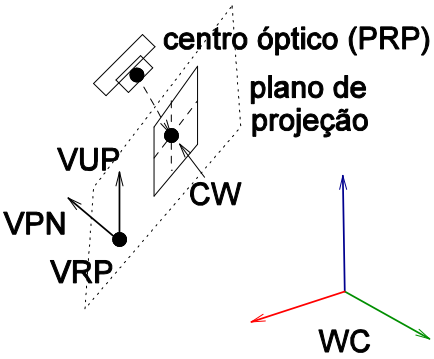
# Modelo de Câmera 2

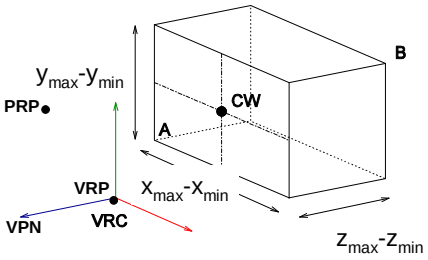



Centrado na câmera

W: posição da câmera
θ :ângulo em relação a (1,0,0,0)
ϕ :ângulo em relação a (0,0,1,0)
r :deslocamento do plano em relação à base da câmera
d :distância focal da lente

# Modelo de Câmera 3

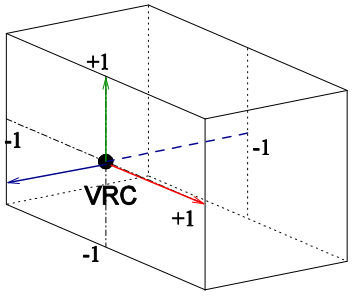


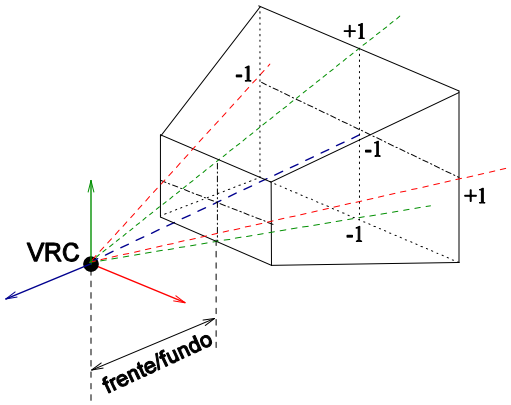

Centrado no plano de projeção

A = ($x_{min}$,$y_{min}$,$z_{min}$)
B=($x_{max}$,$y_{max}$,$z_{max}$)

# Modelo de Espaço Normalizado



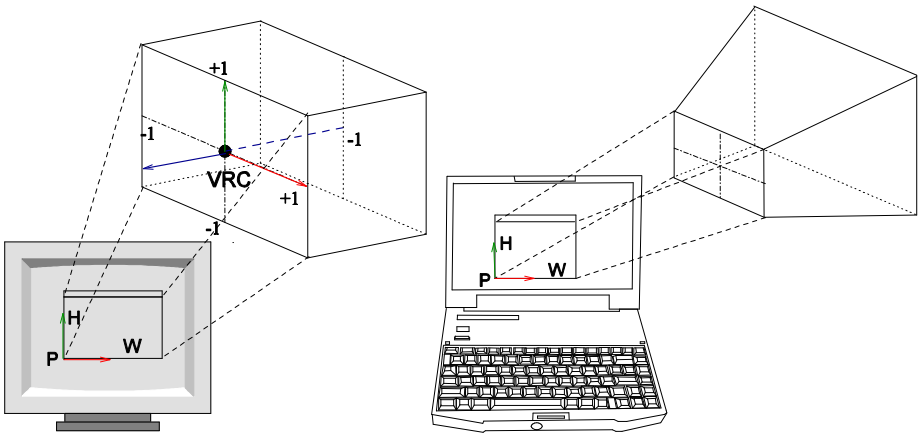

Paralelo




Perspectivo

# Modelo de Dispositivo

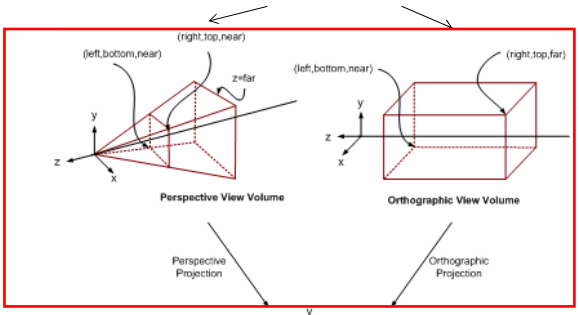
***Viewport***



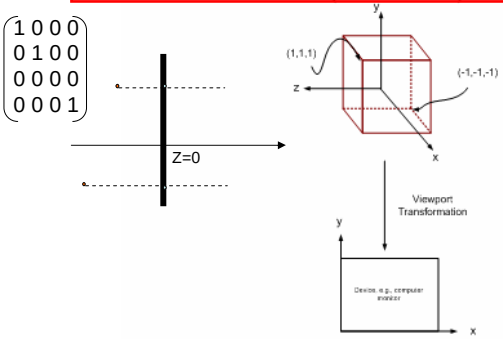

# Universo (WC)

Espaço é deformado para ficar "reto"

Câmera → Normalizado




Espaço da Câmera "Reto" (VRC)



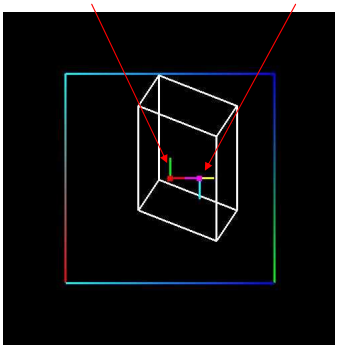

Espaço Normalizado

Espaço da Janela (*Viewport*)

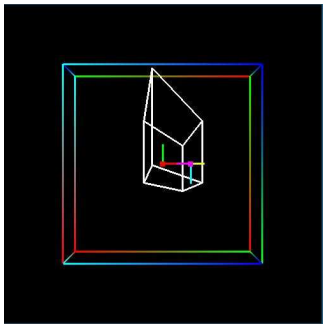
# Projeção Passo a Passo

A partir de um cubo, como chegar à sua projeção

Referencial VRC  Referencial WC



Paralela



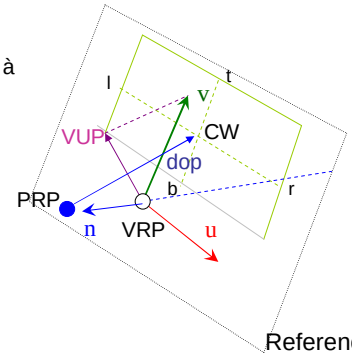
Perspectiva

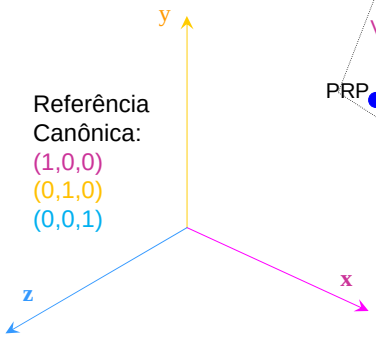
# WC → VRC

## Especificação de VRC

VUP (up): view up vector
VRP: view reference point
VPN (n): view plane normal
PRP: eye
CW (r,l,t,b): centro da janela
dop: CW-PRP (direção de projeção)
profundidade do volume: (F,B) em relação à câmera

$$P' = B_{WC}\, B_{VRC}^{-1} P$$



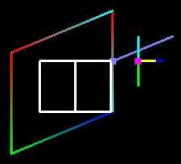




Referência Canônica:
(1,0,0)
(0,1,0)
(0,0,1)

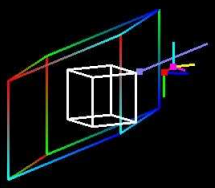
Referencial da Câmera:
u =VUP x n/|VUP x n|
n x u
n = VPN/|VPN|

# Referencial VRC em WC

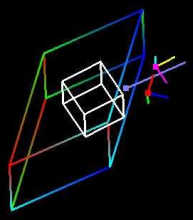
Observe que o referencial WC é a base canônica



Observe que o volume de visão é um paralelepípedo inclinado



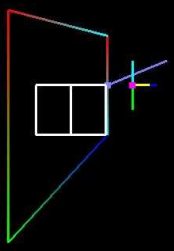
Observe que os referenciais WC e VRC são distintos



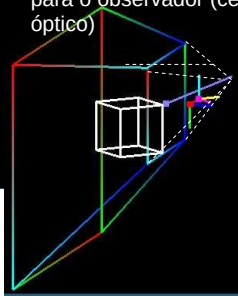
Projeção Paralela
(Várias Vistas)

# Referencial VRC em WC

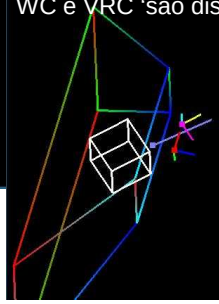
Observe que o referencial WC é a base canônica



Observe que o volume de visão é agora um trapezóide com as arestas convergindo para o observador (centro óptico)

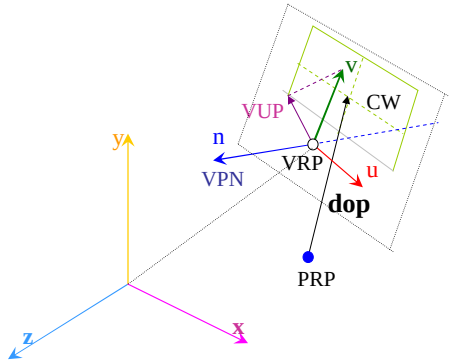


Observe que os referenciais WC e VRC ‘são distintos



Projeção Perspectiva
(Várias Vistas)

# WC → VRC




(1) VRP na origem

$$Tr = \begin{bmatrix} 1 & 0 & 0 & -VRP_x \\ 0 & 1 & 0 & -VRP_y \\ 0 & 0 & 1 & -VRP_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(2) (u,v,n) em vetores base ortonormais

$$\vec{n} = \frac{\overrightarrow{VPN}}{\left|\overrightarrow{VPN}\right|}$$

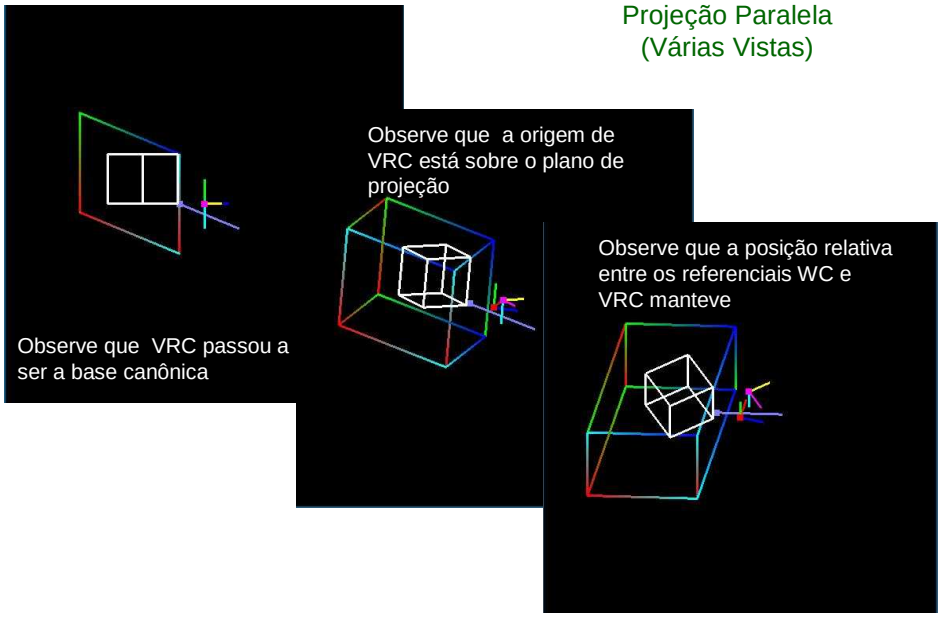
$$\vec{u} = \frac{\overrightarrow{VUP} \times \vec{n}}{\left|\overrightarrow{VUP} \times \vec{n}\right|}$$

$$\vec{v} = \vec{n} \times \vec{u}$$

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

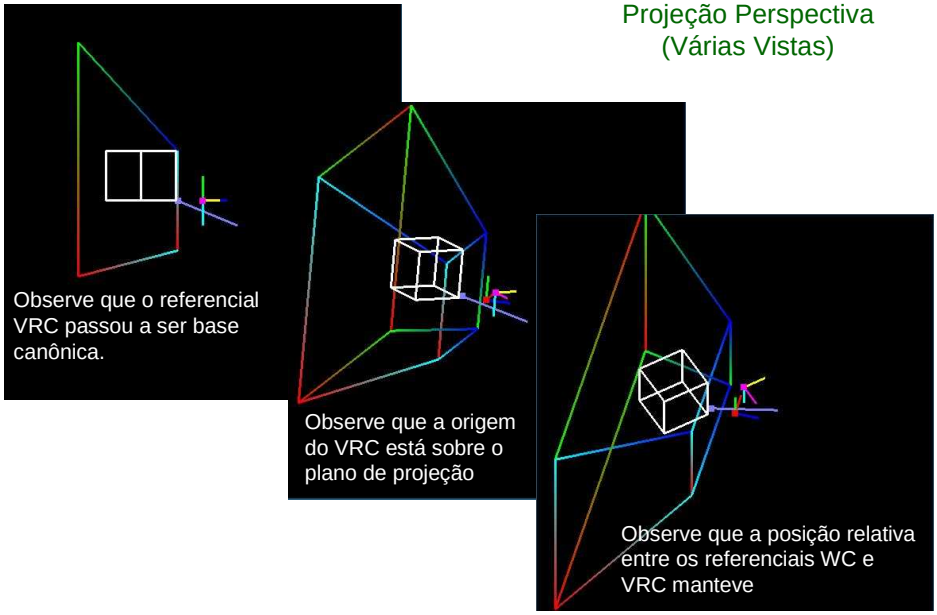
# VRC

Projeção Paralela
(Várias Vistas)



# VRC

Projeção Perspectiva
(Várias Vistas)



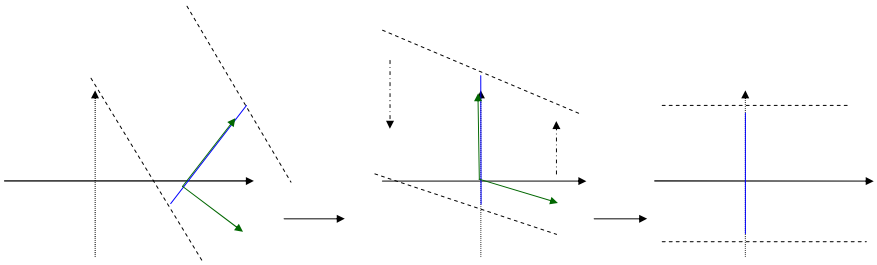

# PRP na origem de VRC

Na projeção paralela o observador está no infinito (raios projetores paralelos), mas na projeção perspectiva, o observador/centro óptico (PRP) está localizado em um ponto finito do espaço. Este ponto deve coincidir com a origem para reduzirmos o nosso problema ao do caso trivial.

$$PRP' = R \; Tr \; PRP$$

$$Tr_{PRP'} = \begin{bmatrix} 1 & 0 & 0 & -PRP'_x \\ 0 & 1 & 0 & -PRP'_y \\ 0 & 0 & 1 & -PRP'_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

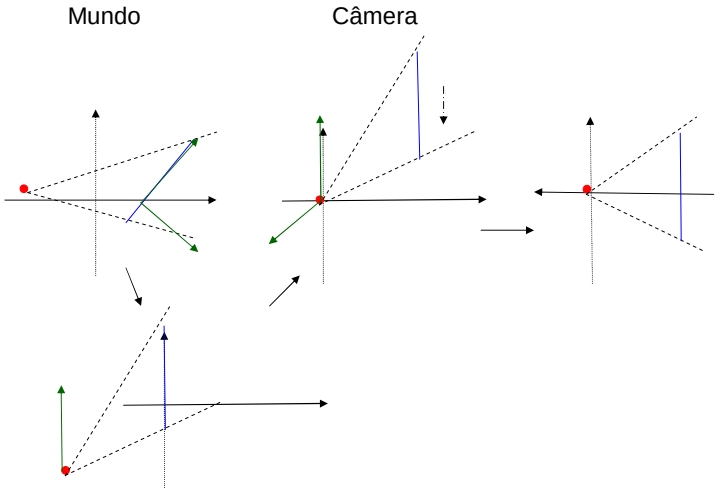
# Deformação do Volume Paralelo
## Cisalhamento



Oblíqua

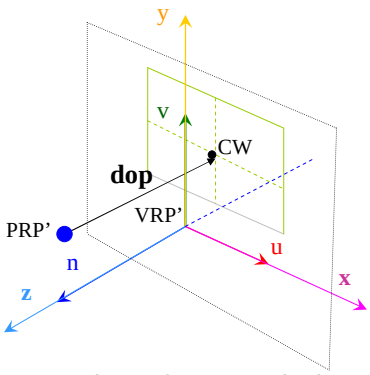
Ortográfica

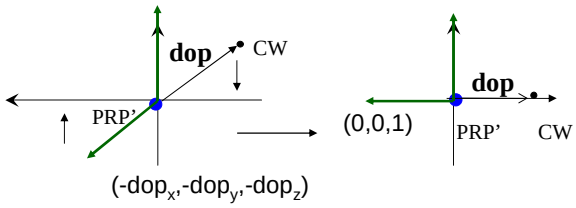
# Deformação do Volume Perspectivo



# Deformação do Volume

## Cisalhamento



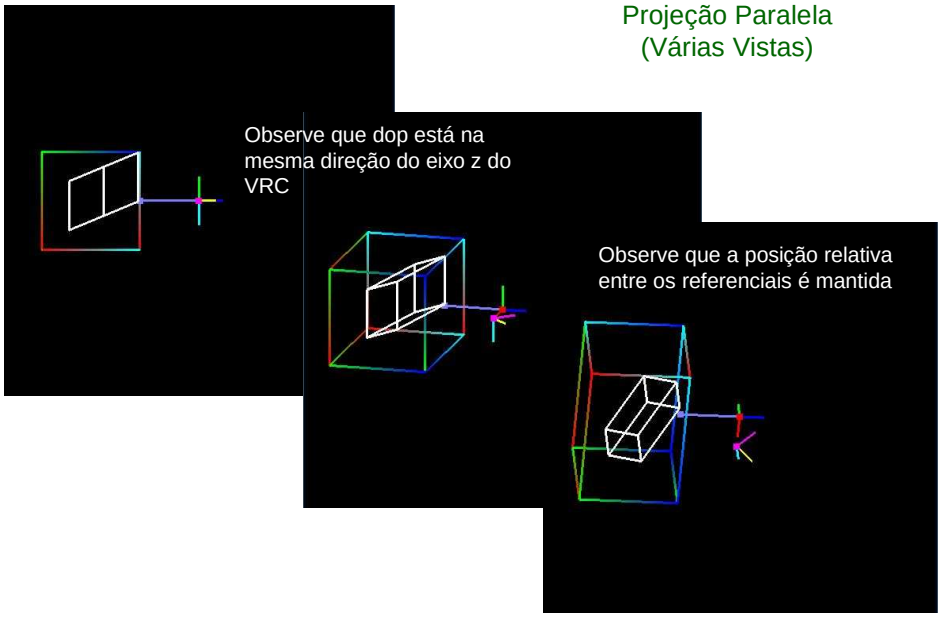

$$\begin{bmatrix} 1 & 0 & -dop_x & 0 \\ 0 & 1 & -dop_y & 0 \\ 0 & 0 & -dop_z & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -dop_z & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

VRP': view reference point in VRC
PRP': projection reference point in VRC
dop: direction of projection
CW: center of window

$$SH = \begin{bmatrix} 1 & 0 & -(dop_x/dop_z) & 0 \\ 0 & 1 & -(dop_y/dop_z) & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

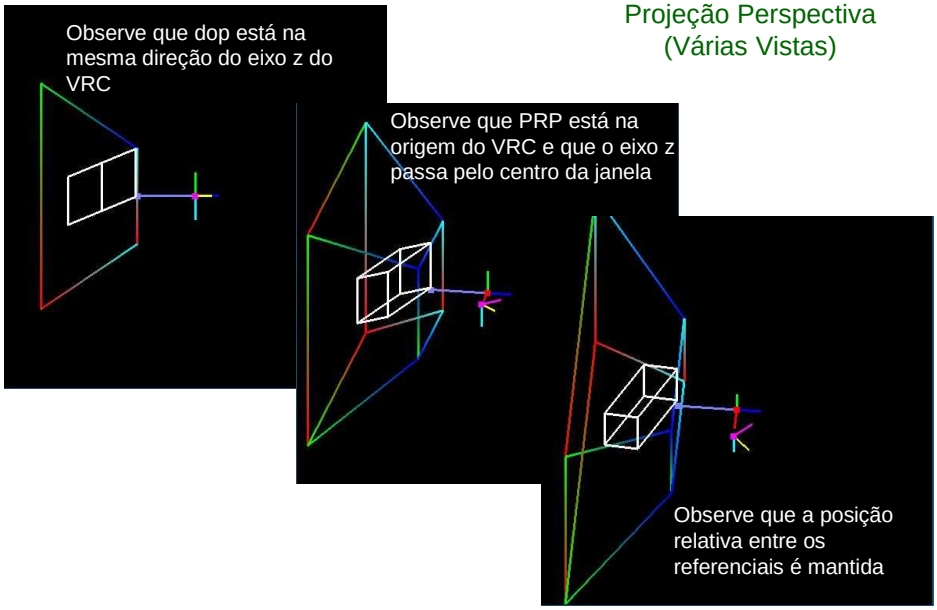
# VRC Cisalhado

Projeção Paralela
(Várias Vistas)



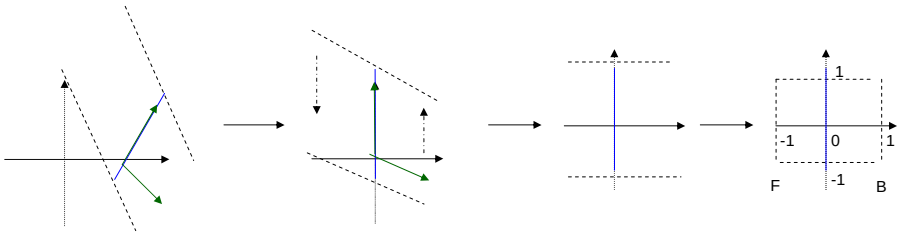
# VRC Cisalhado

Projeção Perspectiva
(Várias Vistas)




# VRC → NDC

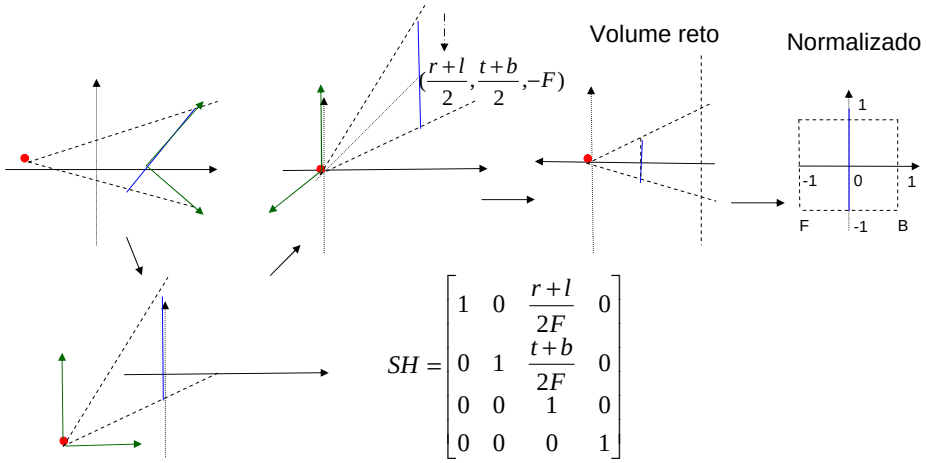
Projeção Paralela




Oblíqua Ortográfica Normalizado

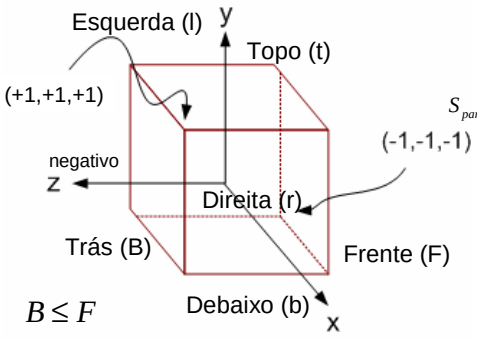
# VRC → NDC

Projeção Perspectiva




$$SH = \begin{bmatrix} 1 & 0 & \frac{r+l}{2F} & 0 \\ 0 & 1 & \frac{t+b}{2F} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

# VRC → NDC (Paralelo)




$B \leq F$

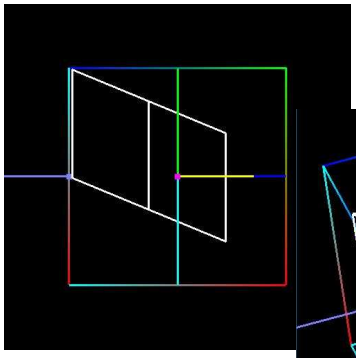
$$S_{par} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & 0 \\ 0 & \frac{2}{t-b} & 0 & 0 \\ 0 & 0 & -\frac{2}{B-F} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -\frac{(r+l)}{2} \\ 0 & 1 & 0 & -\frac{(t+b)}{2} \\ 0 & 0 & 1 & \frac{(B+F)}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & -\frac{2}{B-F} & -\frac{B+F}{B-F} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

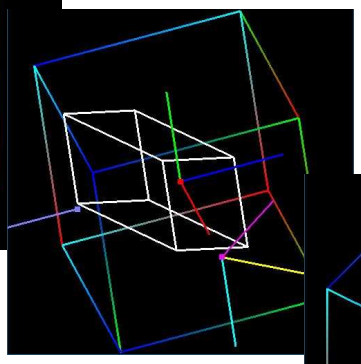
$$\begin{bmatrix} \frac{r+l}{2} & \frac{r+l}{2} & r & r \\ \frac{t+b}{2} & \frac{t+b}{2} & t & -b \\ F & B & F & B \\ 1 & 1 & 1 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

# NDC

Projeção Paralela
(Várias Vistas)

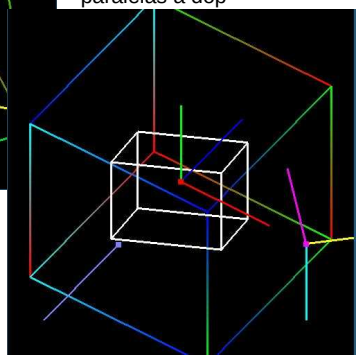


Observe que a relação do volume com o referencial



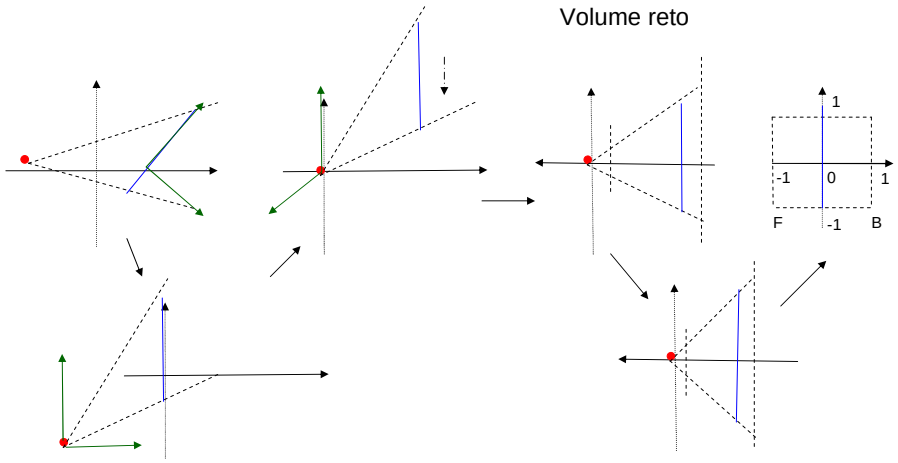
Observe que o volume de visão ficou um cubo centrado na origem do referencial VRC

Observe que o volume de visão tem as arestas paralelas à dop



# VRC → Volume em 45º → NDC (Perspectivo)

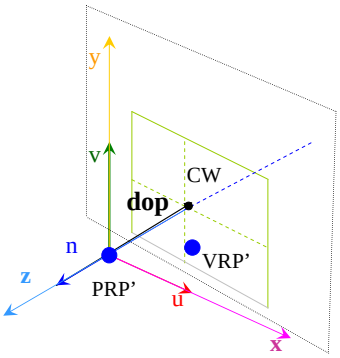
Volume reto

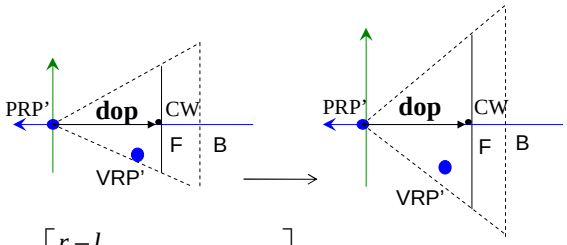



Volume com ângulo $45^0$

# VRC → Volume em 45º

(l,r,b,t,F,B)

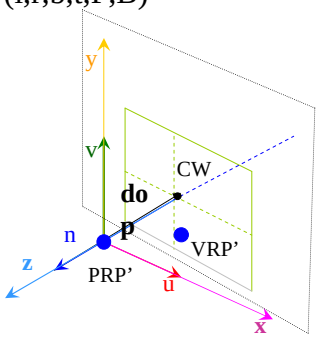


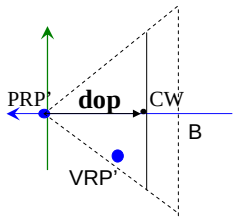




$$\begin{bmatrix} \frac{r-l}{2} & 0 & 0 & 0 \\ 0 & \frac{t-b}{2} & 0 & 0 \\ 0 & 0 & -F & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} F & 0 & 0 & 0 \\ 0 & F & 0 & 0 \\ 0 & 0 & -F & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

# VRC → Volume em 45º

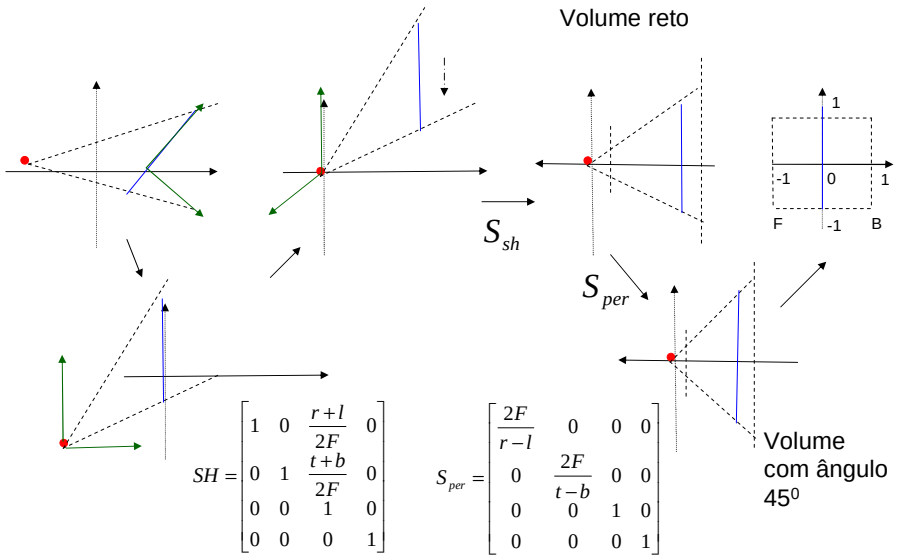
(l,r,b,t,F,B)




$$S_{per} = \begin{bmatrix} \frac{2F}{r-l} & 0 & 0 & 0 \\ 0 & \frac{2F}{t-b} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

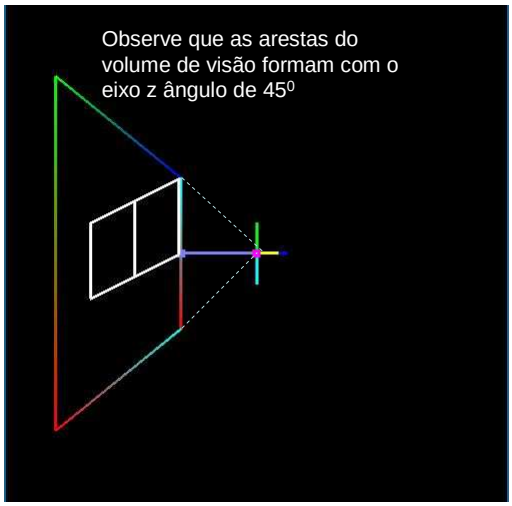
# VRC → Volume em 45º → NDC (Perspectivo)



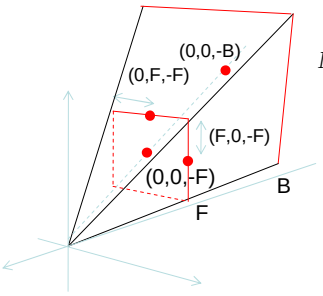

$$SH = \begin{bmatrix} 1 & 0 & \frac{r+l}{2F} & 0 \\ 0 & 1 & \frac{t+b}{2F} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad S_{per} = \begin{bmatrix} \frac{2F}{r-l} & 0 & 0 & 0 \\ 0 & \frac{2F}{t-b} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

# Volume em 45º

Projeção Perspectiva



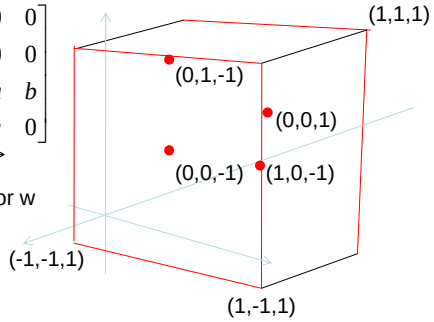
# Volume em 45º → NDC




$$M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & c & 0 \end{bmatrix}$$

+

divisão por w




$$\begin{bmatrix} 0 & F & 0 & 0 \\ 0 & 0 & F & 0 \\ -F & -F & -F & -B \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{-(B+F)}{B-F} & \frac{-2BF}{B-F} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

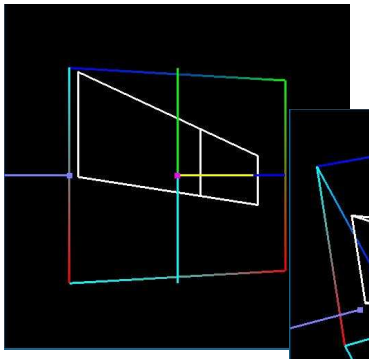
+

divisão por w

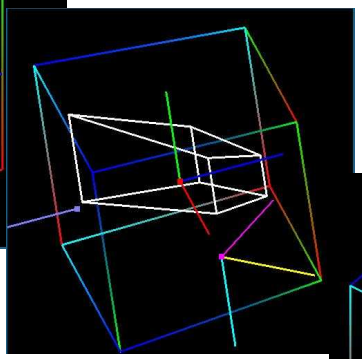
$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

# Normalizado

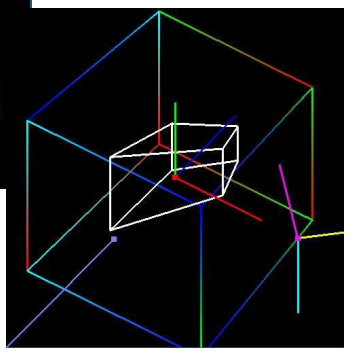
Projeção Perspectivas
(Várias Vistas)



Observe que a relação do volume com o referencial



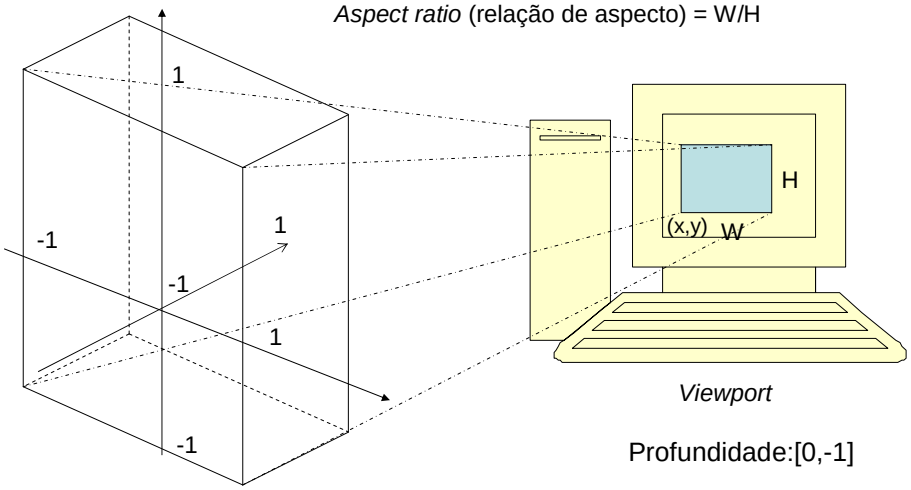
Observe que o volume de visão ficou um cubo centrado na origem do referencial VRC e o cubo dentro do volume ficou distorcido



Observe que PRP foi para "infinito"

# NDC → DC

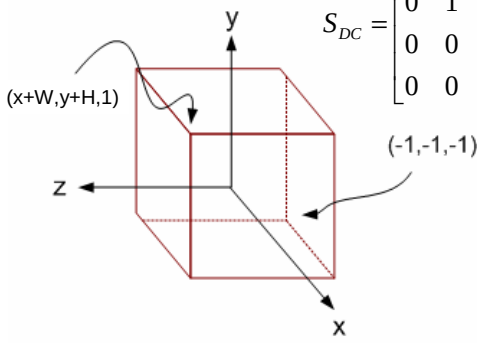
*Aspect ratio* (relação de aspecto) = W/H




*Viewport*

Profundidade:[0,-1]

# DC

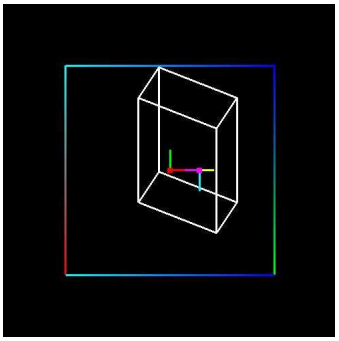



$$S_{DC} = \begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{W}{2} & 0 & 0 & 0 \\ 0 & \frac{H}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix} =$$

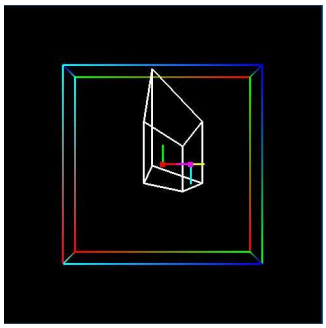
$$\begin{bmatrix} \frac{W}{2} & 0 & 0 & x+\frac{W}{2} \\ 0 & \frac{H}{2} & 0 & y+\frac{H}{2} \\ 0 & 0 & \frac{1}{2} & -\frac{1}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} -1 & 1 & 0 & -1 \\ -1 & 1 & -1 & -1 \\ 1 & 1 & 0 & -1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} x & x+W & x+\frac{W}{2} & x \\ y & y+H & y & y \\ 0 & 0 & -\frac{1}{2} & -1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

# Projeções Paralelas em DC



Paralela



Perspectiva

# Matriz de Transformação

- Projeção Paralela $P_{par} = S_{DC}.S_{par}.SH.R.Tr$
- Projeção Perspectiva $P_{per} = S_{DC}.M.S_{per}.SH.R.Tr$

Transformações lineares

Translação

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & t_x \\ a_{21} & a_{22} & a_{23} & t_y \\ a_{31} & a_{32} & a_{33} & t_z \\ p_x & p_y & p_z & s \end{bmatrix}$$

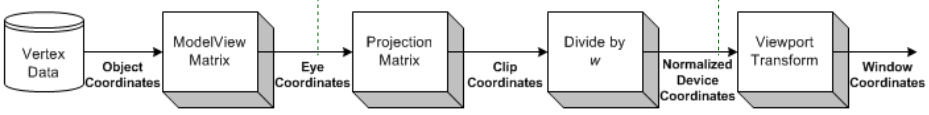
Perspectivo

Número de pontos de fuga: número de elementos p da última linha diferentes de zero

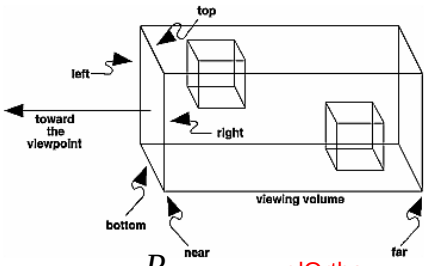
# OpenGL

glMatrixMode

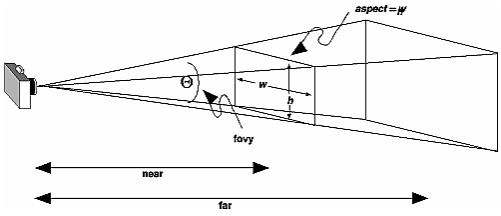
GL_MODELVIEW | GL_PROJECTION $\begin{cases} P_{par,opengl} = S_{par} \\ P_{per,opengl} = M.S_{per}.SH \end{cases}$




$R.Tr$ gluLookAt glViewport




$P_{par,opengl}$ glOrtho



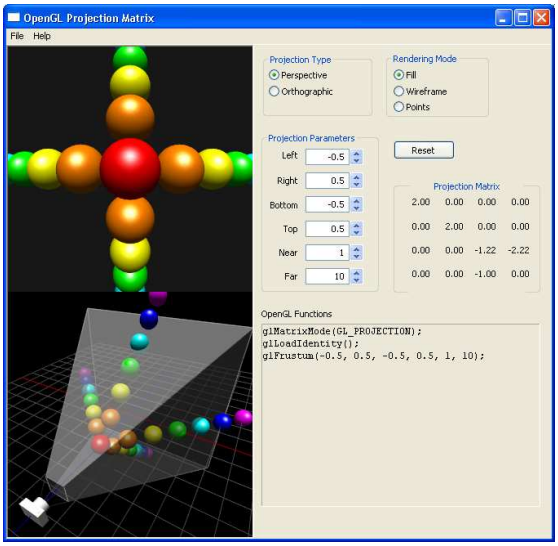

$P_{per,opengl}$ glFrustum

# OpenGL

$$P_{par,opengl} = S_{par} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & -\frac{2}{B-F} & -\frac{B+F}{B-F} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$P_{per,opengl} = M.S_{per}.SH = \begin{bmatrix} \frac{2F}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2F}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{B+F}{B-F} & -\frac{2FB}{B-F} \\ 0 & 0 & -1 & 1 \end{bmatrix}$$

# OpenGL




http://www.songho.ca/opengl/gl_transform.html#projection

# OpenGL

```
void reshape(int w, int h)
{
  glViewport(0, 0, (GLsizei) w, (GLsizei)
    h);
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  if (w <= h)
    glOrtho(-5.0, 5.0, -
   5.0*(GLfloat)h/(GLfloat)w,
    5.0*(GLfloat)h/(GLfloat)w, -5.0, 5.0);
  else
    glOrtho(-5.0*(GLfloat)w/(GLfloat)h,
            5.0*(GLfloat)w/(GLfloat)h, -5.0,
   5.0, -5.0, 5.0);
  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
}
```

```
void init(void)
{
  glClearColor(0.0, 0.0, 0.0, 0.0);
  glShadeModel(GL_FLAT);
  glMap1f(GL_MAP1_VERTEX_3, 0.0, 1.0, 3,
      4,&ctrlpoints[0][0]);
  glEnable(GL_MAP1_VERTEX_3);
  glMapGrid1f(20,0.0,1.0);
}

void display(void)
{
  int i;

  glClear(GL_COLOR_BUFFER_BIT);
  glColor3f(1.0, 1.0, 1.0);
  glBegin(GL_LINE_STRIP);
    for (i = 0; i <= 10; i++)
      glEvalCoord1f((GLfloat) i/10.0);
  glEnd();
  /* The following code displays the control points as dots.
      */
  glPointSize(5.0);
  glColor3f(1.0, 1.0, 0.0);
  glBegin(GL_POINTS);
   for (i = 0; i < 4; i++)
      glVertex3fv(&ctrlpoints[i][0]);
  glEnd();
  glFlush();

}
```

# OpenGL

```
void display(void) {
    glClear (GL_COLOR_BUFFER_BIT);
    glColor3f (1.0, 1.0, 1.0);
    glLoadIdentity (); /* clear the matrix */
    /* viewing transformation */
    gluLookAt (0.0, 0.0, 5.0, 0.0, 0.0, 0.0,
    0.0, 1.0, 0.0);
    glScalef (1.0, 2.0, 1.0); /* modeling
    transformation */
    glutWireCube (1.0);
    glFlush ();
}
void reshape (int w, int h) {
    glViewport (0, 0, (GLsizei) w,
    (GLsizei) h);
    glMatrixMode (GL_PROJECTION);
    glLoadIdentity ();
    glFrustum (-1.0, 1.0, -1.0, 1.0, 1.5,
    20.0);
    glMatrixMode (GL_MODELVIEW);
}
```

```
void init(void) {
    glClearColor (0.0, 0.0, 0.0, 0.0);
    glShadeModel (GL_FLAT);
}

int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode
    (GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize (500, 500);
    glutInitWindowPosition (100, 100);
    glutCreateWindow (argv[0]);
    init ();
    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
    glutMainLoop();
    return 0;
}
```

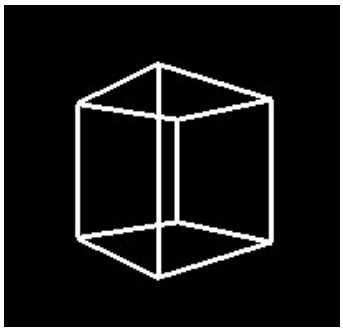
# OpenGL

```
void init(void) {
    /* Enable a single OpenGL light. */
    glLightfv(GL_LIGHT0, GL_DIFFUSE,
    light_diffuse);
    glLightfv(GL_LIGHT0,
    GL_POSITION, light_position);
    glEnable(GL_LIGHT0);
    glEnable(GL_LIGHTING);
    /* Use depth buffering for hidden
    surface elimination. */
    glEnable(GL_DEPTH_TEST);
    /* Setup the view of the cube. */
    glMatrixMode(GL_PROJECTION);
    gluPerspective( /* field of view in
    degree */ 40.0, /* aspect ratio */ 1.0,
    /* Z near */ 1.0, /* Z far */ 10.0);
    glMatrixMode(GL_MODELVIEW);
    gluLookAt(0.0, 0.0, 5.0, /* eye is at
    (0,0,5) */ 0.0, 0.0, 0.0, /* center is at
    (0,0,0) */ 0.0, 1.0, 0.); /* up is in
    positive Y direction */
}
```

```
/* Adjust cube position to be
    asthetic angle. */

void display(void) {
    glClear(GL_COLOR_BUFFE
    R_BIT |
    GL_DEPTH_BUFFER_BIT);
      /* Adjust cube position to
    be asthetic angle. */
     glTranslatef(0.0, 0.0, -1.0);
     glRotatef(60, 1.0, 0.0, 0.0);
     glPushMatrix();
     glRotatef(-20, 0.0, 0.0, 1.0);
     drawBox();
     glPopMatrix();
     glTranslatef(1.0, -1.0, 0.5);
     drawBox();
     glutSwapBuffers();
}
```

# OpenGL: 2 Pontos de Fuga



glGetDoublev(GL_PROJECTION_MATRIX, projection);

```
0.667  0.000  0.000  0.000
0.000  0.667  0.000  0.000
0.000  0.000 -1.500 -2.500
0.000  0.000 -1.000  0.000
```

glGetDoublev(GL_MODELVIEW_MATRIX, modelview);

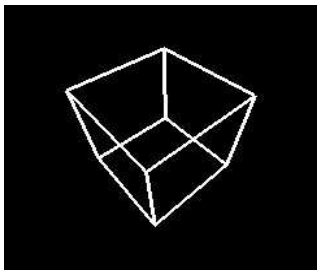
```
-0.643  0.000 -0.766  0.000
 0.000  1.000  0.000  0.000
 0.766  0.000 -0.643 -2.000
 0.000  0.000  0.000  1.000
```

projection* modelview =

```
-0.429  0.000 -0.511  0.000
 0.000  0.667  0.000  0.000
-1.149  0.000  0.964  0.500
-0.766  0.000  0.643  2.000
```

2 pontos de fuga

# OpenGL: 3 Pontos de Fuga



glGetDoublev(GL_PROJECTION_MATRIX, projection);

```
0.667  0.000  0.000  0.000
0.000  0.667  0.000  0.000
0.000  0.000 -1.500 -2.500
0.000  0.000 -1.000  0.000
```

glGetDoublev(GL_MODELVIEW_MATRIX, modelview);

```
0.762 -0.030  0.647  0.000
0.510  0.643 -0.571  0.000
-0.399  0.765  0.506 -2.000
0.000  0.000  0.000  1.000
```

projection* modelview =

```
0.508 -0.020  0.431  0.000
0.340  0.429 -0.381  0.000
0.598 -1.147 -0.759  0.500
0.399 -0.765 -0.506  2.000
```

3 pontos de fuga