

# Plan

## 4 Synthèse d'image

- Présentation
- OpenGL
- De la scène 3D à l'image 2D
- Rendu

# Principe

# Principe

- 1 création de la scène :

# Principe

- 1 création de la scène :
  - création d'objets de base à l'aide de maillages,

# Principe

## ① création de la scène :

- création d'objets de base à l'aide de maillages,
- composition d'une scène complexe à l'aide des objets de base, et de transformations géométriques.

# Principe

## ① création de la scène :

- création d'objets de base à l'aide de maillages,
- composition d'une scène complexe à l'aide des objets de base, et de transformations géométriques.

## ② définition de la caméra :

# Principe

## 1 création de la scène :

- création d'objets de base à l'aide de maillages,
- composition d'une scène complexe à l'aide des objets de base, et de transformations géométriques.

## 2 définition de la caméra :

- positionnement de celle-ci par rapport à la scène,

# Principe

## 1 création de la scène :

- création d'objets de base à l'aide de maillages,
- composition d'une scène complexe à l'aide des objets de base, et de transformations géométriques.

## 2 définition de la caméra :

- positionnement de celle-ci par rapport à la scène,
- choix de la projection et limitation de la profondeur.



# Principe

- 1 création de la scène :
  - création d'objets de base à l'aide de maillages,
  - composition d'une scène complexe à l'aide des objets de base, et de transformations géométriques.
- 2 définition de la caméra :
  - positionnement de celle-ci par rapport à la scène,
  - choix de la projection et limitation de la profondeur.
- 3 définition de la taille de l'image finale

# Objets de base

# Objets de base

Vertex (point ou sommet)

**Exemple** : définir le point de coordonnées (2, 3, 0)

```
glVertex3d(2.0, 3.0, 0.0);
```

ou bien

```
GLdouble S[3] = {2.0, 3.0, 0.0};  
glVertex3dv(S);
```

# Objets de base

Line (segment de droite)

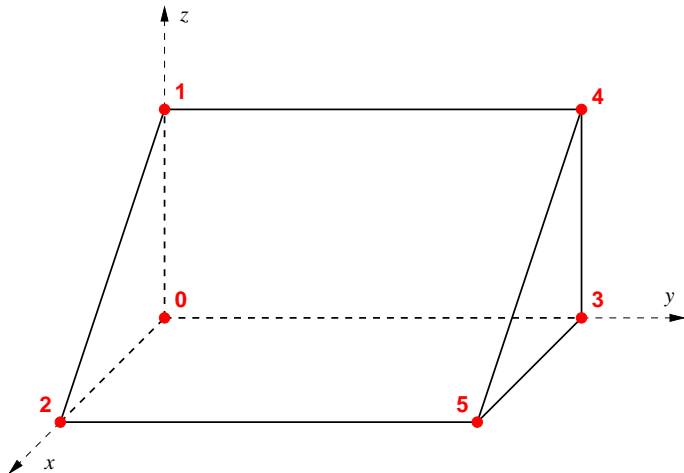
**Exemple** : définir le segment du point  $(-1, 2, 5)$  au point  $(0, 4, 1)$

```
glBegin ( GL_LINES );  
    glVertex3d ( -1.0 , 2.0 , 5.0 );  
    glVertex3d (  0.0 , 4.0 , 1.0 );  
glEnd ();
```

# Objets de base

Triangle / quadrangle

## Exemple 1 : prisme



# Objets de base

Triangle / quadrangle

## Exemple 1 : prisme

Sommet	x	y	z
<b>0</b>	0	0	0
<b>1</b>	0	0	2
<b>2</b>	2	0	0
<b>3</b>	0	3	0
<b>4</b>	0	3	2
<b>5</b>	2	3	0

Face	nS	sommets
<b>0</b>	3	<b>2 1 0</b>
<b>1</b>	3	<b>3 4 5</b>
<b>2</b>	4	<b>4 1 2 5</b>
<b>3</b>	4	<b>5 2 0 3</b>
<b>4</b>	4	<b>3 0 1 4</b>

# Objets de base

Triangle / quadrangle

## Exemple 1 : prisme

```
void prisme()  
{  
    GLdouble S[6][3] = {  
        {0.0,0.0,0.0}, {0.0,0.0,2.0}, {2.0,0.0,0.0},  
        {0.0,3.0,0.0}, {0.0,3.0,2.0}, {2.0,3.0,0.0}};  
  
    /* les deux triangles */  
    glBegin(GL_TRIANGLES);  
        glVertex3dv(S[2]); glVertex3dv(S[1]); glVertex3dv(S[0]);  
        glVertex3dv(S[3]); glVertex3dv(S[4]); glVertex3dv(S[5]);  
    glEnd();  
  
    ...  
}
```

# Objets de base

Triangle / quadrangle

## Exemple 1 : prisme

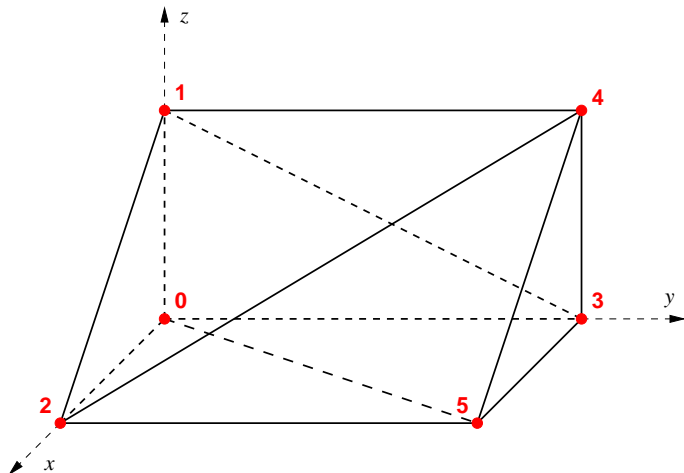
```
void prisme()  
{  
    ...  
  
    /* les trois quadrangles */  
    glBegin(GL_QUADS);  
        glVertex3dv(S[4]); glVertex3dv(S[1]);  
        glVertex3dv(S[2]); glVertex3dv(S[5]);  
    glVertex3dv(S[5]); glVertex3dv(S[2]);  
        glVertex3dv(S[0]); glVertex3dv(S[3]);  
    glVertex3dv(S[3]); glVertex3dv(S[0]);  
        glVertex3dv(S[1]); glVertex3dv(S[4]);  
    glEnd();  
}
```



# Objets de base

Triangle / quadrangle

## Exemple 2 : prisme triangulé



# Objets de base

Triangle / quadrangle

## Exemple 2 : prisme triangulé

Sommet	x	y	z
<b>0</b>	0	0	0
<b>1</b>	0	0	2
<b>2</b>	2	0	0
<b>3</b>	0	3	0
<b>4</b>	0	3	2
<b>5</b>	2	3	0

Face	sommets
<b>0</b>	<b>2 1 0</b>
<b>1</b>	<b>3 4 5</b>
<b>2</b>	<b>4 1 2</b>
<b>3</b>	<b>4 2 5</b>
<b>4</b>	<b>5 2 0</b>
<b>5</b>	<b>5 0 3</b>
<b>6</b>	<b>3 0 1</b>
<b>7</b>	<b>3 1 4</b>

# Objets de base

## Triangle / quadrangle

### Exemple 2 : prisme triangulé

```
void prisme_triangule()
{
    GLdouble S[6][3] = {
        {0.0,0.0,0.0}, {0.0,0.0,2.0}, {2.0,0.0,0.0},
        {0.0,3.0,0.0}, {0.0,3.0,2.0}, {2.0,3.0,0.0}};
    int T[8][3] = {{2,1,0}, {3,4,5}, {4,1,2}, {4,2,5},
                  {5,2,0}, {5,0,3}, {3,0,1}, {3,1,4}};

    // les différents triangles
    glBegin(GL_TRIANGLES);
    for (int i=0; i<8; i++)
        // un triangle
        for (int j=0; j<3; j++)
            glVertex3dv(S[T[i]][j]);
    glEnd();
}
```

# Transformations géométriques

# Transformations géométriques

## Opérations de base - exemple dans le plan

# Transformations géométriques

Opérations de base - exemple dans le plan

## Homothétie de facteur $r$

$$\text{point } \begin{pmatrix} x \\ y \end{pmatrix} \longrightarrow \text{point } \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} rx \\ ry \end{pmatrix} = \begin{pmatrix} r & 0 \\ 0 & r \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

$$\text{opération linéaire : } \begin{pmatrix} x' \\ y' \end{pmatrix} = M \begin{pmatrix} x \\ y \end{pmatrix}$$

# Transformations géométriques

Opérations de base - exemple dans le plan

## Rotation d'angle $a$

$$\begin{aligned} \text{point } \begin{pmatrix} x \\ y \end{pmatrix} &\longrightarrow \text{point } \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x \cos(a) - y \sin(a) \\ x \sin(a) + y \cos(a) \end{pmatrix} \\ &= \begin{pmatrix} \cos(a) & -\sin(a) \\ \sin(a) & \cos(a) \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \end{aligned}$$

$$\text{opération linéaire : } \begin{pmatrix} x' \\ y' \end{pmatrix} = M \begin{pmatrix} x \\ y \end{pmatrix}$$

# Transformations géométriques

Opérations de base - exemple dans le plan

**Translation du vecteur**  $\begin{pmatrix} x_T \\ y_T \end{pmatrix}$

$$\text{point} \begin{pmatrix} x \\ y \end{pmatrix} \longrightarrow \text{point} \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x + x_T \\ y + y_T \end{pmatrix}$$

opération non linéaire car impossible d'écrire  $\begin{pmatrix} x' \\ y' \end{pmatrix} = M \begin{pmatrix} x \\ y \end{pmatrix}$



# Transformations géométriques

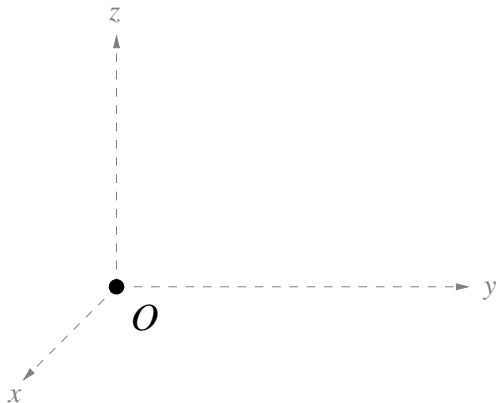
Utilisation des coordonnées homogènes

Principe dans le plan  $\mathbb{R}^2$

# Transformations géométriques

## Utilisation des coordonnées homogènes

Principe dans le plan  $\mathbb{R}^2$

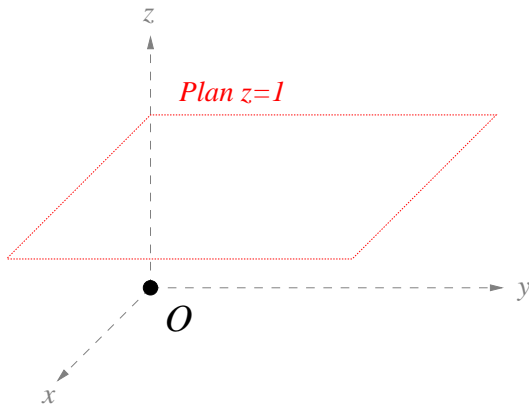


Espace des coordonnées homogènes  $\mathbb{R}^3$

# Transformations géométriques

## Utilisation des coordonnées homogènes

Principe dans le plan  $\mathbb{R}^2$

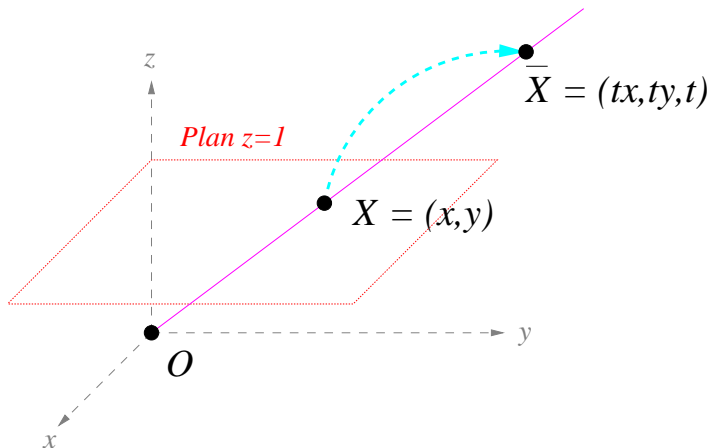


Espace réel  $\mathbb{R}^2$  : plan  $\{(x, y)\} \equiv \text{plan } \{(x, y, z), z = 1\} \subset \mathbb{R}^3$

# Transformations géométriques

Utilisation des coordonnées homogènes

Principe dans le plan  $\mathbb{R}^2$

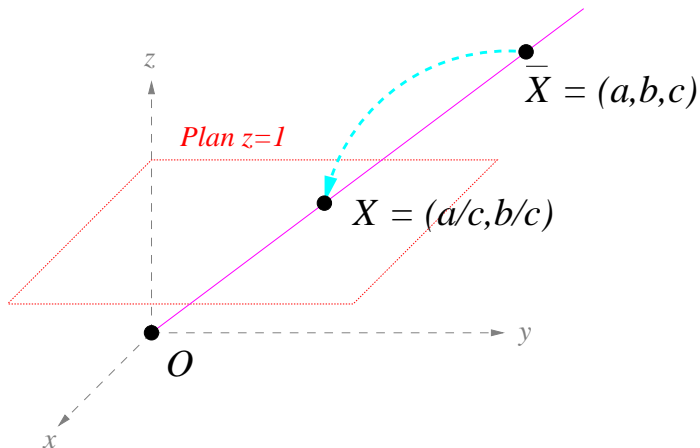


$$X = (x, y) \in \mathbb{R}^2 \rightarrow \bar{X} = (t x, t y, t) \in \mathbb{R}^3 \text{ (avec } t > 0\text{)}$$

# Transformations géométriques

## Utilisation des coordonnées homogènes

Principe dans le plan  $\mathbb{R}^2$



$$\bar{X} = (a, b, c) \in \mathbb{R}^3 \rightarrow X = (a/c, b/c) \in \mathbb{R}^2$$

# Utilisation des coordonnées homogènes

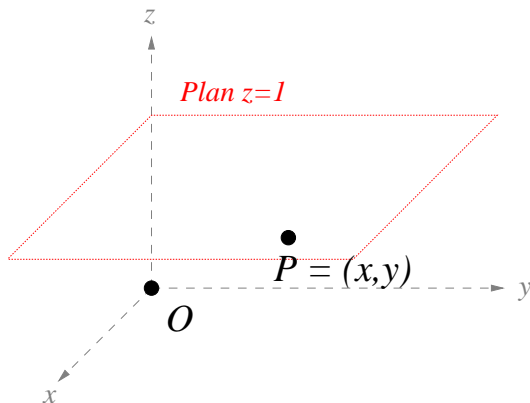
Opération géométrique  $T$  dans  $\mathbb{R}^2$

Principe dans le plan  $\mathbb{R}^2$

# Utilisation des coordonnées homogènes

Opération géométrique  $T$  dans  $\mathbb{R}^2$

Principe dans le plan  $\mathbb{R}^2$

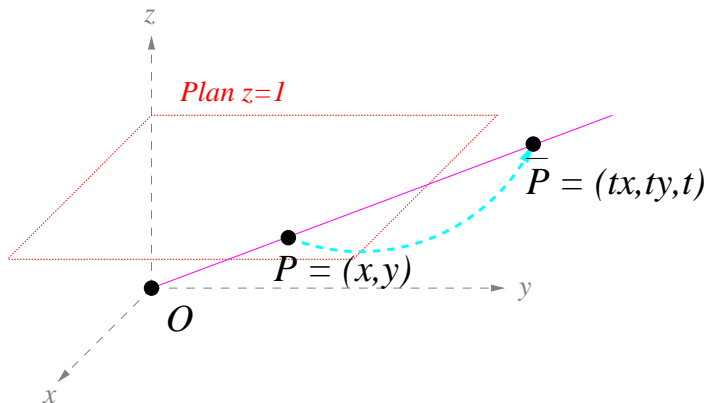


$$P = (x, y) \in \mathbb{R}^2$$

# Utilisation des coordonnées homogènes

Opération géométrique  $T$  dans  $\mathbb{R}^2$

Principe dans le plan  $\mathbb{R}^2$



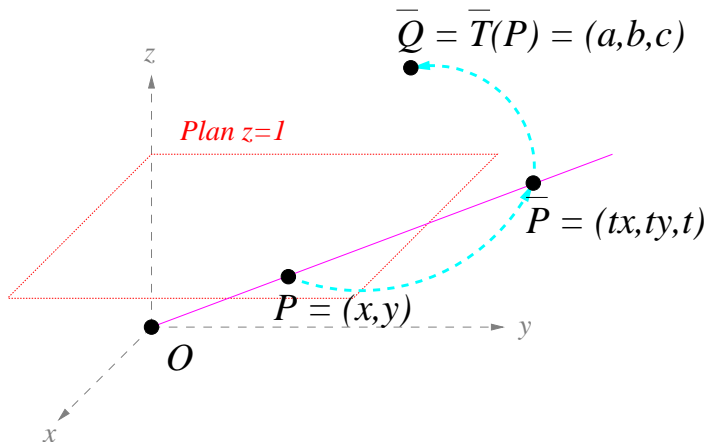
$$P \equiv \bar{P} = (t x, t y, t) \in \mathbb{R}^3$$



# Utilisation des coordonnées homogènes

Opération géométrique  $T$  dans  $\mathbb{R}^2$

Principe dans le plan  $\mathbb{R}^2$

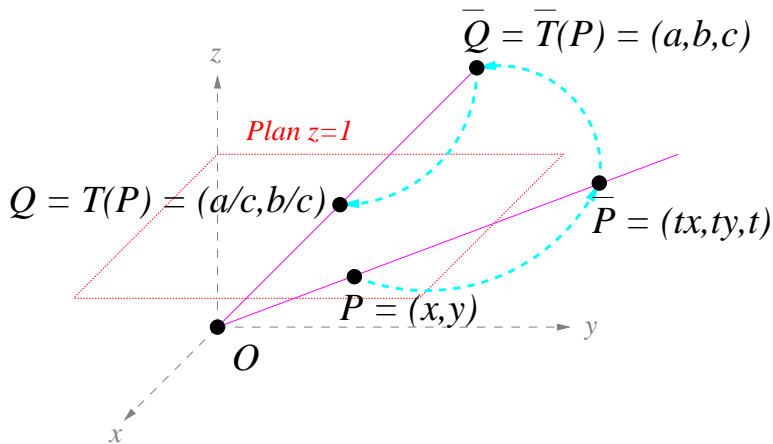


$$\rightarrow \bar{Q} = \bar{T}(\bar{P}) = (a, b, c) \in \mathbb{R}^3$$

# Utilisation des coordonnées homogènes

Opération géométrique  $T$  dans  $\mathbb{R}^2$

Principe dans le plan  $\mathbb{R}^2$



$$\bar{Q} \equiv Q = (a/c, b/c) \in \mathbb{R}^2, Q = T(P)$$

# Utilisation des coordonnées homogènes

Opération géométrique  $T$  dans  $\mathbb{R}^2$

**Homothétie de rapports  $r_x$  en  $x$  et  $r_y$  en  $y$**

$$P = \begin{pmatrix} x \\ y \end{pmatrix} \rightarrow \bar{P} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

↓

$$Q = \begin{pmatrix} r_x x \\ r_y y \\ 1 \end{pmatrix} \leftarrow \bar{Q} = \begin{pmatrix} r_x & 0 & 0 \\ 0 & r_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} r_x x \\ r_y y \\ 1 \end{pmatrix}$$

# Utilisation des coordonnées homogènes

Opération géométrique  $T$  dans  $\mathbb{R}^2$

## Rotation d'angle $a$

$$P = \begin{pmatrix} x \\ y \end{pmatrix} \rightarrow \bar{P} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

↓

$$\bar{Q} = \begin{pmatrix} \cos(a) & -\sin(a) & 0 \\ \sin(a) & \cos(a) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

# Utilisation des coordonnées homogènes

Opération géométrique  $T$  dans  $\mathbb{R}^2$

## Rotation d'angle $a$

$$P = \begin{pmatrix} x \\ y \end{pmatrix} \quad \rightarrow \quad \bar{P} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

↓

$$Q = \begin{pmatrix} x \cos(a) - y \sin(a) \\ x \sin(a) + y \cos(a) \end{pmatrix} \quad \leftarrow \quad \bar{Q} = \begin{pmatrix} x \cos(a) - y \sin(a) \\ x \sin(a) + y \cos(a) \\ 1 \end{pmatrix}$$

# Utilisation des coordonnées homogènes

Opération géométrique  $T$  dans  $\mathbb{R}^2$

**Translation de vecteur**  $\begin{pmatrix} x_T \\ y_T \end{pmatrix}$

$$P = \begin{pmatrix} x \\ y \end{pmatrix} \rightarrow \bar{P} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

↓

$$Q = \begin{pmatrix} x + x_T \\ y + y_T \end{pmatrix} \leftarrow \bar{Q} = \begin{pmatrix} 1 & 0 & x_T \\ 0 & 1 & y_T \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x + x_T \\ y + y_T \\ 1 \end{pmatrix}$$

# Opérations géométriques dans $\mathbb{R}^3$

# Opérations géométriques dans $\mathbb{R}^3$

Opération géométrique dans  $\mathbb{R}^3$



# Opérations géométriques dans $\mathbb{R}^3$

Opération géométrique dans  $\mathbb{R}^3$

équivalent à

# Opérations géométriques dans $\mathbb{R}^3$

Opération géométrique dans  $\mathbb{R}^3$

équivalent à

Opération linéaire dans  $\mathbb{R}^4$  en coordonnées homogènes

Matrice carrée associée  $\bar{M}$  de dimension 4

# Opérations géométriques dans $\mathbb{R}^3$

**Homothétie de rapports  $r_x$  en  $x$ ,  $r_y$  en  $y$  et  $r_z$  en  $z$**

$$\bar{M} = \begin{pmatrix} r_x & 0 & 0 & 0 \\ 0 & r_y & 0 & 0 \\ 0 & 0 & r_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

# Opérations géométriques en 3D

**Rotation d'angle  $a$  autour de l'axe  $Ox$**

$$\bar{M} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(a) & -\sin(a) & 0 \\ 0 & \sin(a) & \cos(a) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

# Opérations géométriques en 3D

**Rotation d'angle  $a$  autour de l'axe  $Oy$**

$$\bar{M} = \begin{pmatrix} \cos(a) & 0 & \sin(a) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(a) & 0 & \cos(a) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

# Opérations géométriques en 3D

**Rotation d'angle  $a$  autour de l'axe  $Oz$**

$$\bar{M} = \begin{pmatrix} \cos(a) & -\sin(a) & 0 & 0 \\ \sin(a) & \cos(a) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

# Opérations géométriques en 3D

**Opération linéaire de matrice associée**  $M = \begin{pmatrix} m_{1,1} & m_{1,2} & m_{1,3} \\ m_{2,1} & m_{2,2} & m_{2,3} \\ m_{3,1} & m_{3,2} & m_{3,3} \end{pmatrix}$   
(exemples : changement de base, projection sur un plan, symétrie, ...)

$$\bar{M} = \begin{pmatrix} m_{1,1} & m_{1,2} & m_{1,3} & 0 \\ m_{2,1} & m_{2,2} & m_{2,3} & 0 \\ m_{3,1} & m_{3,2} & m_{3,3} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

# Opérations géométriques en 3D

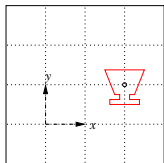
**Translation de vecteur**  $\begin{pmatrix} x_T \\ y_T \\ z_T \end{pmatrix}$

$$\bar{M} = \begin{pmatrix} 1 & 0 & 0 & x_T \\ 0 & 1 & 0 & y_T \\ 0 & 0 & 1 & z_T \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



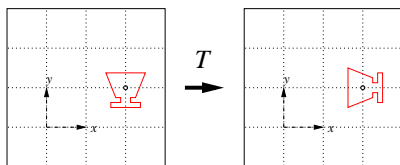
# Exemple de composition d'opérations géométriques

# Exemple de composition d'opérations géométriques



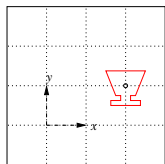
Objet initial

# Exemple de composition d'opérations géométriques

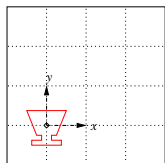


Opération  $T$  : rotation de  $90^\circ$  autour du point  $(2, 1)$

# Exemple de composition d'opérations géométriques



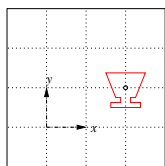
$\downarrow T_1$



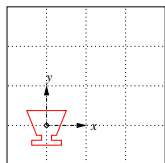
$T$  composition des opérations

$T_1$  : translation de vecteur  $(-2, -1)$  : matrice associée  $\bar{M}_1$

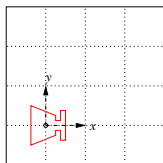
# Exemple de composition d'opérations géométriques



$T_1$



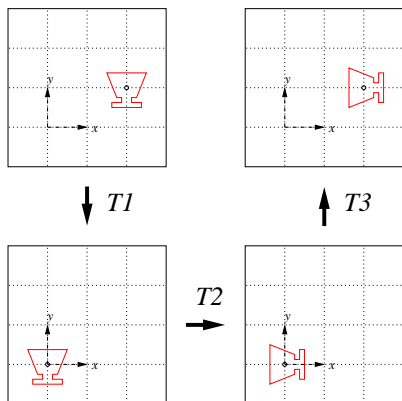
$T_2$



$T$  composition des opérations

$T_2$  : rotation de  $90^\circ$  : matrice associée  $\bar{M}_2$

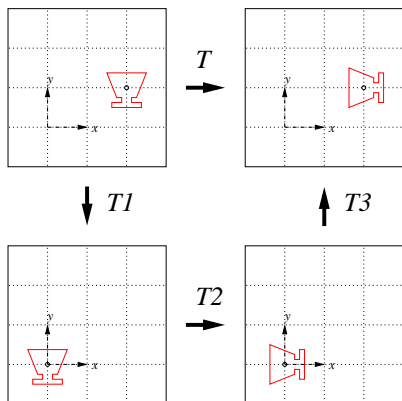
# Exemple de composition d'opérations géométriques



$T$  composition des opérations

$T_3$  : translation de vecteur (2, 1) : matrice associée  $\bar{M}_3$

# Exemple de composition d'opérations géométriques



$T$  composition des opérations

$T = T_3 \circ T_2 \circ T_1$  : matrice associée, produit  $\bar{M}_3 \bar{M}_2 \bar{M}_1$

## Exemple de composition d'opérations géométriques

Opération  $T_1$  : matrice associée  $\bar{M}_1 = \begin{pmatrix} 1 & 0 & -2 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{pmatrix}$

Opération  $T_2$  : matrice associée  $\bar{M}_2 = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$

Opération  $T_3$  : matrice associée  $\bar{M}_3 = \begin{pmatrix} 1 & 0 & 2 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$

⇒ Opération  $T = T_3 \circ T_2 \circ T_1$  :

matrice associée  $\bar{M} = \bar{M}_3 \bar{M}_2 \bar{M}_1 = \begin{pmatrix} 0 & -1 & 3 \\ 1 & 0 & -1 \\ 0 & 0 & 1 \end{pmatrix}$



## Exemple de composition d'opérations géométriques

La traduction algorithmique :

```
debut_transformation()  
  translation(2,1) // transformation T3  
  
  rotation(90) // transformation T2  
  
  translation(-2,-1) // transformation T1  
  
dessin_objet()  
fin_transformation()
```

## Exemple de composition d'opérations géométriques

Le code OpenGL correspondant :

```
glPushMatrix();  
  glTranslated(2.0,1.0,0.0); // transformation T3  
  // translation de (2,1,0)  
  
  glRotated(90.0,0.0,0.0,1.0); // transformation T2  
  // rotation de 90° autour de l'axe (0,0,1)  
  
  glTranslated(-2.0,-1.0,0.0); // transformation T1  
  // translation de (-2,-1,0)  
  
dessin_objet();  
glPopMatrix();
```

# Exemple de composition d'opérations géométriques

OpenGL : utilisation de piles de matrices afin de mémoriser les différentes opérations géométriques, en particulier :

- une pile de matrices pour la scène 3D (`GL_MODELVIEW`)
- une pile de matrices pour la projection utilisée par la caméra (`GL_PROJECTION`)

# Exemple de composition d'opérations géométriques

Spécification d'une pile de matrices :

`glMatrixMode(GL_MODELVIEW)` ou `glMatrixMode(GL_PROJECTION)`

# Exemple de composition d'opérations géométriques

Initialiser la pile avec la matrice identité :

```
glLoadIdentity()
```

(→ la matrice courante de la pile devient la matrice identité

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

## Exemple de composition d'opérations géométriques

Commencer une opération géométrique :

```
glPushMatrix()
```

(→ la matrice courante de la pile est dupliquée en haut de la pile)

Terminer une opération géométrique :

```
glPopMatrix()
```

(→ la matrice en haut de la pile est supprimée)

## Exemple de composition d'opérations géométriques

Effectuer une opération géométrique : par ex. `glRotate`, `glTranslate`, `glScale`, ...

(→ la matrice courante  $\bar{M}_C$  de la pile devient  $\bar{M}_C \bar{M}$  avec  $\bar{M}$  matrice de l'opération géométrique)

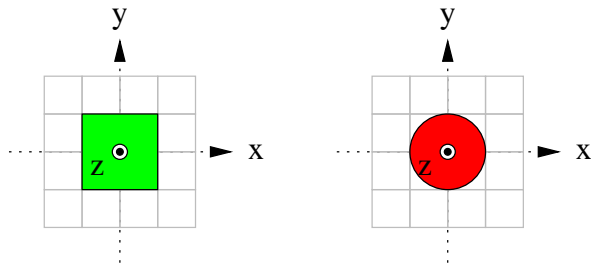
# Composition d'une scène à partir de primitives

## Le wagon



# Composition d'une scène à partir de primitives

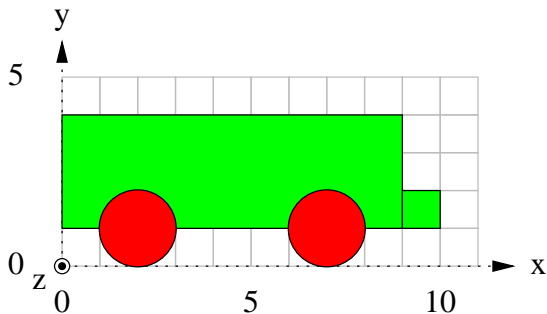
## Le wagon



Les objets de base

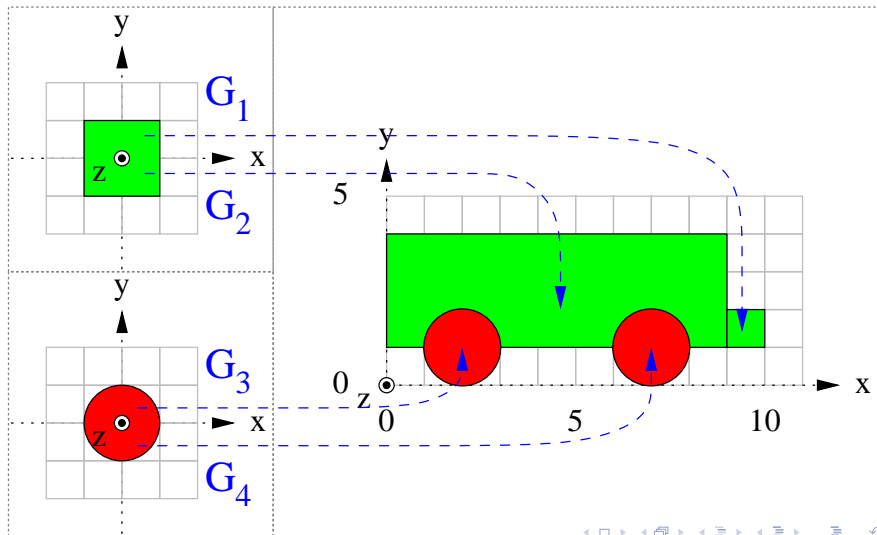
# Composition d'une scène à partir de primitives

## Le wagon



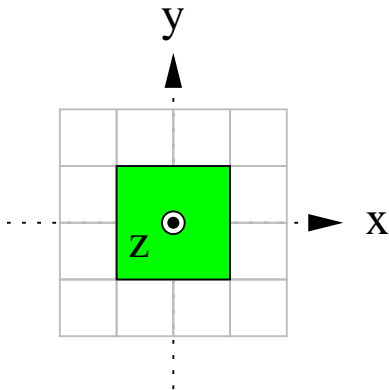
# Composition d'une scène à partir de primitives

## Le wagon



# Composition d'une scène à partir de primitives

## Le carré unité



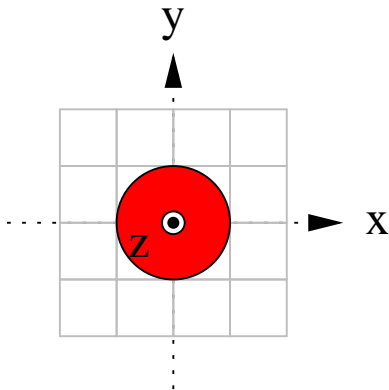
# Composition d'une scène à partir de primitives

## Le carré unité

```
void carre()  
{  
    // remplissage du carre  
    glColor3d(0,1,0); // vert  
    glBegin(GL_QUADS);  
        glVertex3d(-1.0, -1.0,0.0); glVertex3d(-1.0, 1.0,0.0);  
        glVertex3d( 1.0, 1.0,0.0); glVertex3d( 1.0, -1.0,0.0);  
    glEnd();  
  
    // tracé du bord du carre  
    glColor3d(0,0,0); // noir  
    glBegin(GL_LINE_LOOP);  
        glVertex3d(-1.0, -1.0,0.0); glVertex3d(-1.0, 1.0,0.0);  
        glVertex3d( 1.0, 1.0,0.0); glVertex3d( 1.0, -1.0,0.0);  
    glEnd();  
}
```

# Composition d'une scène à partir de primitives

## Le cercle unité



# Composition d'une scène à partir de primitives

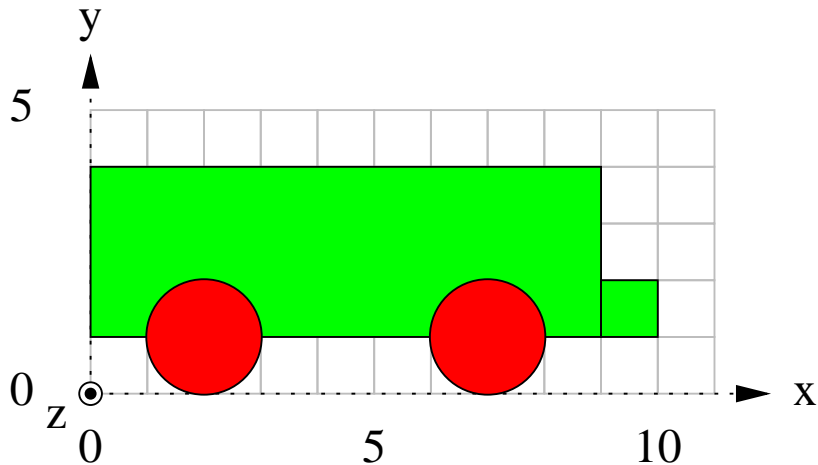
## Le cercle unité

```
void cercle ()
{
    // remplissage du cercle
    glColor3d(1,0,0); // rouge
    glBegin(GL_POLYGON);
        for (int i=0; i<100; i++)
            glVertex3d(cos(0.02*i*M_PI), sin(0.02*i*M_PI), 0.01);
    glEnd();

    // tracé du bord du cercle
    glColor3d(0,0,0); // noir
    glBegin(GL_LINE_LOOP);
        for (int i=0; i<100; i++)
            glVertex3d(cos(0.02*i*M_PI), sin(0.02*i*M_PI), 0.01);
    glEnd();
}
```

# Composition d'une scène à partir de primitives

## Le wagon





# Composition d'une scène à partir de primitives

## Le wagon

```
void wagon()  
{  
    glPushMatrix(); // la partie 1  
        glTranslated(9.5,1.5,0.0);  
        glScaled(0.5,0.5,1.0);  
        carre();  
    glPopMatrix();  
  
    glPushMatrix(); // la partie 2  
        glTranslated(4.5,2.5,0.0);  
        glScaled(4.5,1.5,1.0);  
        carre();  
    glPopMatrix();  
  
    // ...  
}
```

# Composition d'une scène à partir de primitives

## Le wagon

```
// ...  
  
glPushMatrix(); // la partie 3  
    glTranslated(2.0,1.0,0.0);  
    cercle();  
glPopMatrix();  
  
glPushMatrix(); // la partie 4  
    glTranslated(7.0,1.0,0.0);  
    cercle();  
glPopMatrix();  
}
```

# Composition d'une scène à partir de primitives

## La scène complète

```
void dessin_scene()
{
    // ...

    // placement d'un wagon en position (X,Y,Z)
    glPushMatrix();
        glTranslated(X,Y,Z);
        wagon();
    glPopMatrix();

    // ...
}
```

# L'affichage de la scène en OpenGL

```
void dessin ()
{
    // effacer le buffer-image en le remplissant en blanc
    glClearColor (1.0 ,1.0 ,1.0 ,1.0);
    glClear (GL_COLOR_BUFFER_BIT);

    // initialiser les transformations pour la scène
    glMatrixMode (GL_MODELVIEW);
    glLoadIdentity ();

    // dessiner la scène
    dessin_scene ();

    // afficher le buffer – mode simple buffer
    glFlush ();
}
```

# L'affichage de la scène en OpenGL

```
void dessin ()
{
    // effacer le buffer-image en le remplissant en blanc
    glClearColor (1.0 ,1.0 ,1.0 ,1.0);
    glClear (GL_COLOR_BUFFER_BIT);

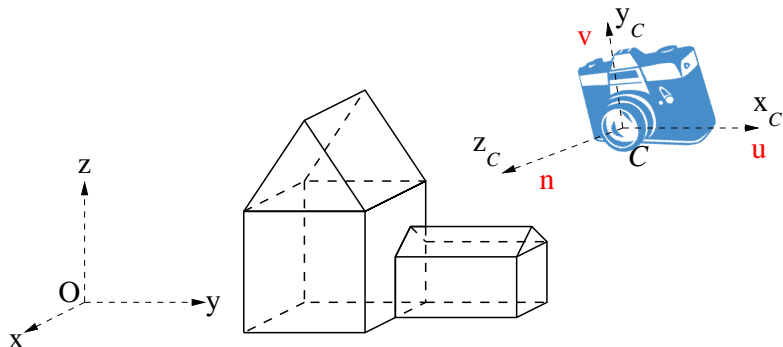
    // initialiser les transformations pour la scène
    glMatrixMode (GL_MODELVIEW);
    glLoadIdentity ();

    // dessiner la scène
    dessin_scene ();

    // afficher le buffer – mode double buffer
    glutSwapBuffers ();
}
```

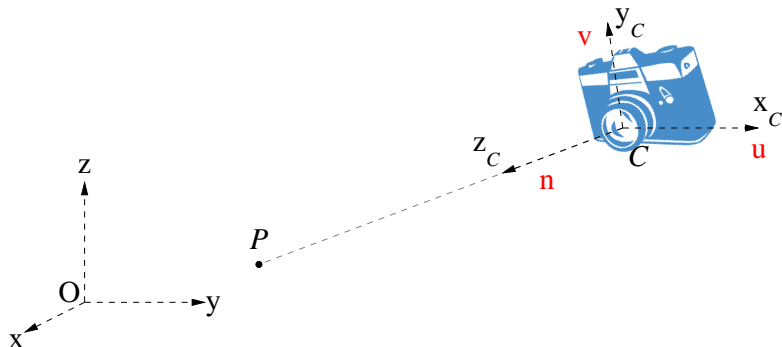
# Placement de la caméra

# Placement de la caméra



```
gluLookAt(Cx,Cy,Cz , Px,Py,Pz , vx,vy,vz)
```

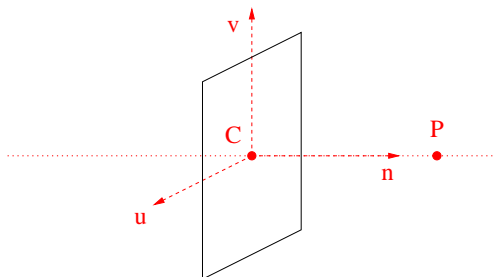
# Placement de la caméra



```
gluLookAt(Cx,Cy,Cz , Px,Py,Pz , vx,vy,vz)
```



# Placement de la caméra



Plan de projection de l'image finale  
 $\Pi$  défini par le point  $C$  et les vecteurs  $(u, v)$

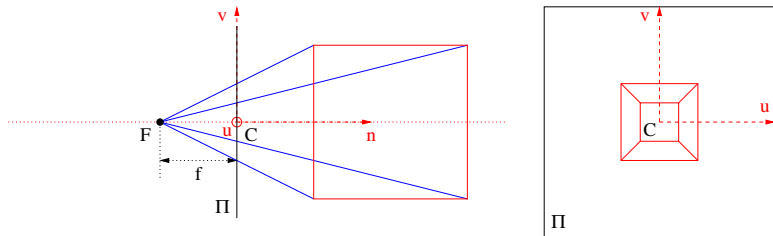
# Choix de la projection

# Choix de la projection

## Projection perspective

# Choix de la projection

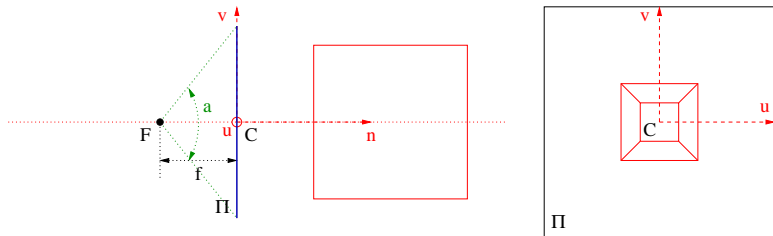
## Projection perspective



$C$  : position de la caméra /  $F$  : point focal  
 Petite focale  $f$  / grand angle  $a$

# Choix de la projection

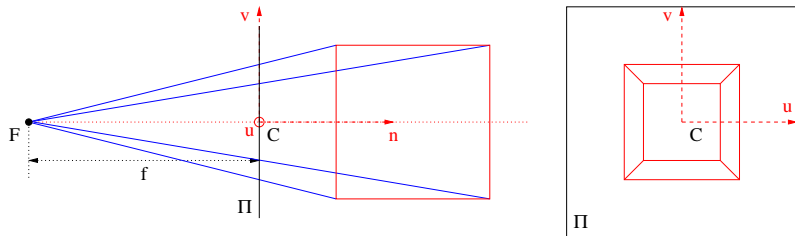
## Projection perspective



$C$  : position de la caméra /  $F$  : point focal  
 Petite focale  $f$  / grand angle  $a$

# Choix de la projection

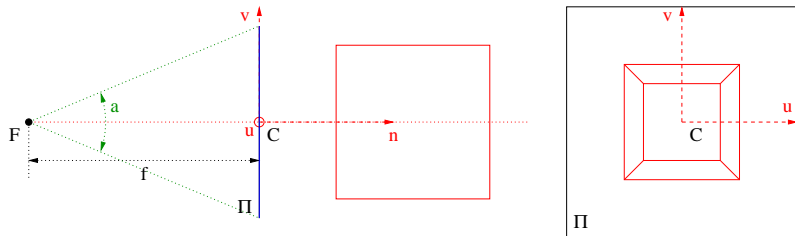
## Projection perspective



$C$  : position de la caméra /  $F$  : point focal  
 Grande focale  $f$  / petit angle  $\alpha$

# Choix de la projection

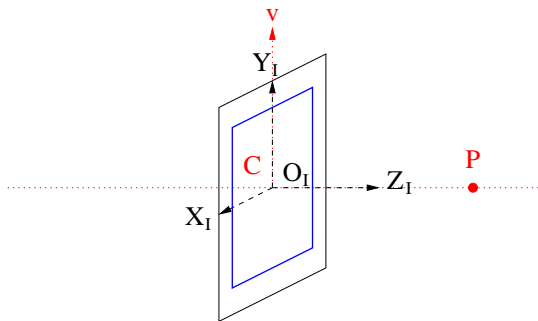
## Projection perspective



$C$  : position de la caméra /  $F$  : point focal  
 Grande focale  $f$  / petit angle  $a$

# Choix de la projection

Projection perspective - limitation du volume vu

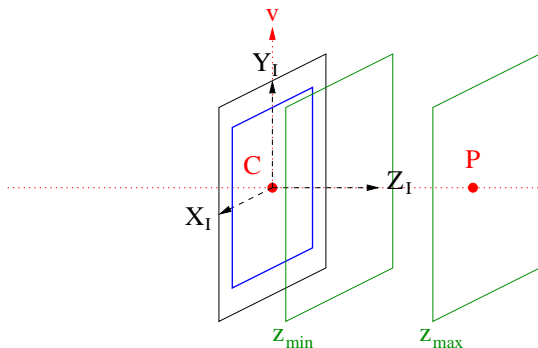


Plan de projection et image finale



# Choix de la projection

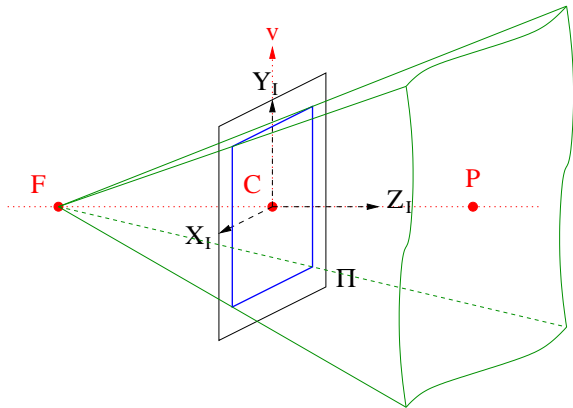
Projection perspective - limitation du volume vu



Plans de coupe

# Choix de la projection

Projection perspective - limitation du volume vu



Projection perspective

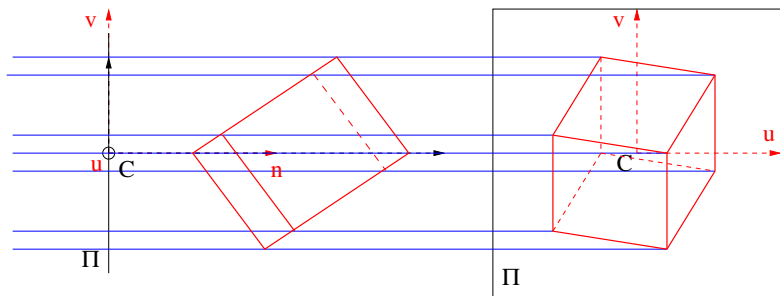


# Choix de la projection

Projection orthographique (point focal à l'infini)

# Choix de la projection

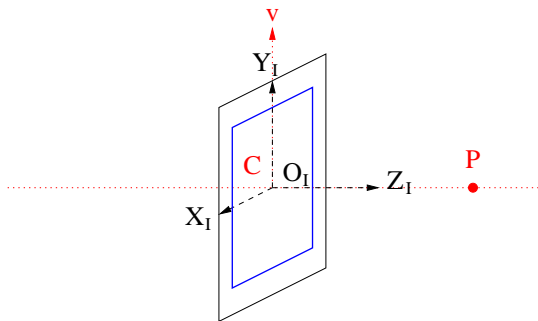
Projection orthographique (point focal à l'infini)



Point de fuite à l'infini : focale  $f$  infinie

# Choix de la projection

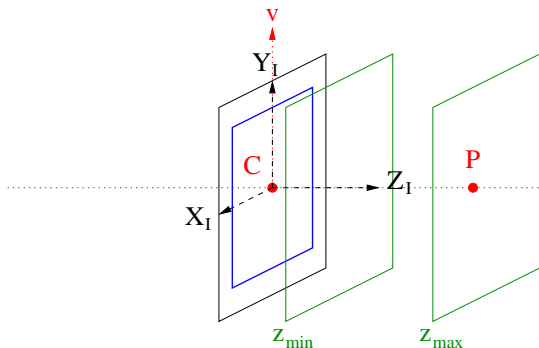
Projection orthographique - limitation du volume vu



Plan de projection et image finale

# Choix de la projection

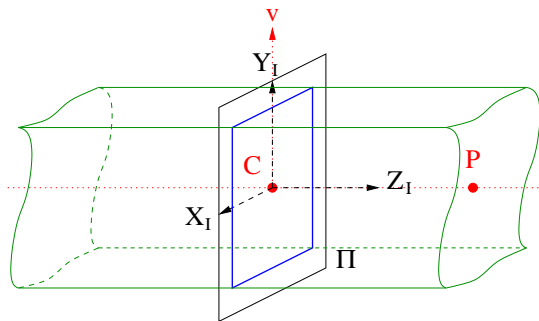
Projection orthographique - limitation du volume vu



Plans de coupe

# Choix de la projection

Projection orthographique - limitation du volume vu

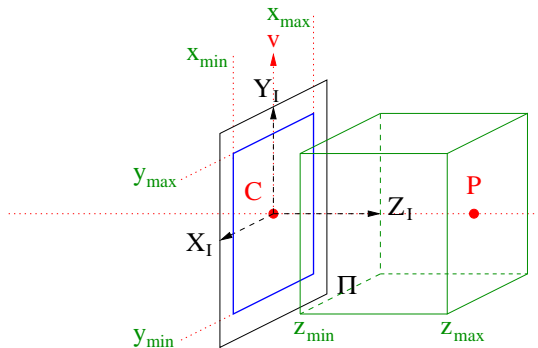


Projection orthographique



# Choix de la projection

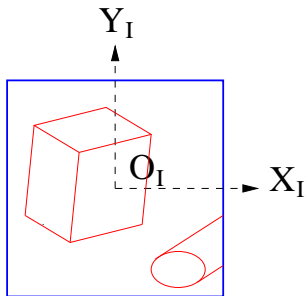
## Projection orthographique - limitation du volume vu



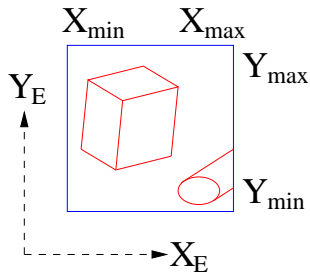
Volume correspondant - projection orthographique  
`glOrtho(xmin,xmax , ymin,ymax , zmin,zmax)`

# Affichage de l'image finale

# Affichage de l'image finale



**Plan de projection**



**Ecran**

`glViewport(Xmin, Ymin, Xmax-Xmin, Ymax-Ymin)`

# Définition du système (caméra/projection/image finale)

cas projection perspective

```
void redimensionnement(int L_IMAGE, int H_IMAGE)
{
    // dimension de l'image finale
    glViewport (0, 0, L_IMAGE, H_IMAGE);
    double ratio_f = (double)L_IMAGE/((double)H_IMAGE);

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();

    // choix de la projection perspective
    gluPerspective(angle_camera, ratio_f, z_min, z_max);

    // positionnement de la caméra
    gluLookAt(Cx, Cy, Cz, Px, Py, Pz, vx, vy, vz);
}
```

# Définition du système (caméra/projection/image finale)

cas projection orthographique

```
void redimensionnement(int L_IMAGE, int H_IMAGE)
{
    // dimension de l'image finale
    glViewport (0, 0, L_IMAGE, H_IMAGE);

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();

    // choix de la projection orthographique
    glOrtho(x_min, x_max, y_min, y_max, z_min, z_max);

    // positionnement de la caméra
    gluLookAt(Cx, Cy, Cz, Px, Py, Pz, vx, vy, vz);
}
```

# Définition du système (caméra/projection/image finale)

# Définition du système (caméra/projection/image finale)

- coordonnées dans le repère *scène* :  
 $C_x, C_y, C_z, P_x, P_y, P_z, v_x, v_y, v_z$

# Définition du système (caméra/projection/image finale)

- coordonnées dans le repère *scène* :  
 $C_x, C_y, C_z, P_x, P_y, P_z, v_x, v_y, v_z$
- coordonnées dans le repère *caméra* :  
 $x_{\min}, x_{\max}, y_{\min}, y_{\max}, z_{\min}, z_{\max}$



# Définition du système (caméra/projection/image finale)

- coordonnées dans le repère *scène* :  
 $C_x, C_y, C_z, P_x, P_y, P_z, v_x, v_y, v_z$
- coordonnées dans le repère *caméra* :  
 $x_{\min}, x_{\max}, y_{\min}, y_{\max}, z_{\min}, z_{\max}$
- coordonnées dans le repère *image* :  
 $L\_IMAGE, H\_IMAGE$

# Définition du système (caméra/projection/image finale)

- coordonnées dans le repère *scène* :  
Cx, Cy, Cz, Px, Py, Pz, vx, vy, vz
- coordonnées dans le repère *caméra* :  
x\_min, x\_max, y\_min, y\_max, z\_min, z\_max
- coordonnées dans le repère *image* :  
L\_IMAGE, H\_IMAGE

avec de préférence

$$\frac{x_{\max} - x_{\min}}{y_{\max} - y_{\min}} = \frac{L\_IMAGE}{H\_IMAGE}$$

