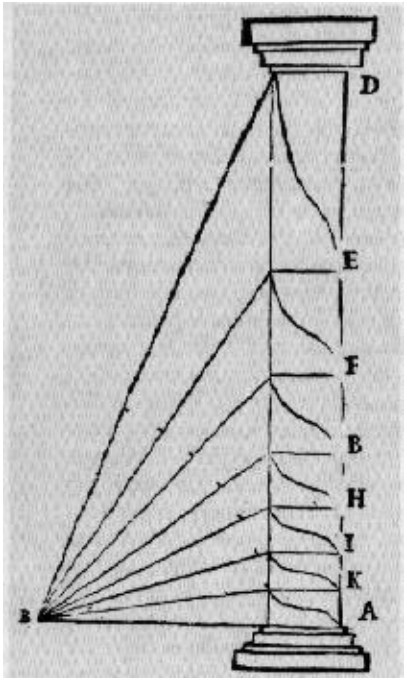
Synthèse d'images

(6) Projections 2D & clôturage



La colonne Trajane à Rome

Athanase Kircher, 1649

Plan de l'exposé :

1- Projections planaires

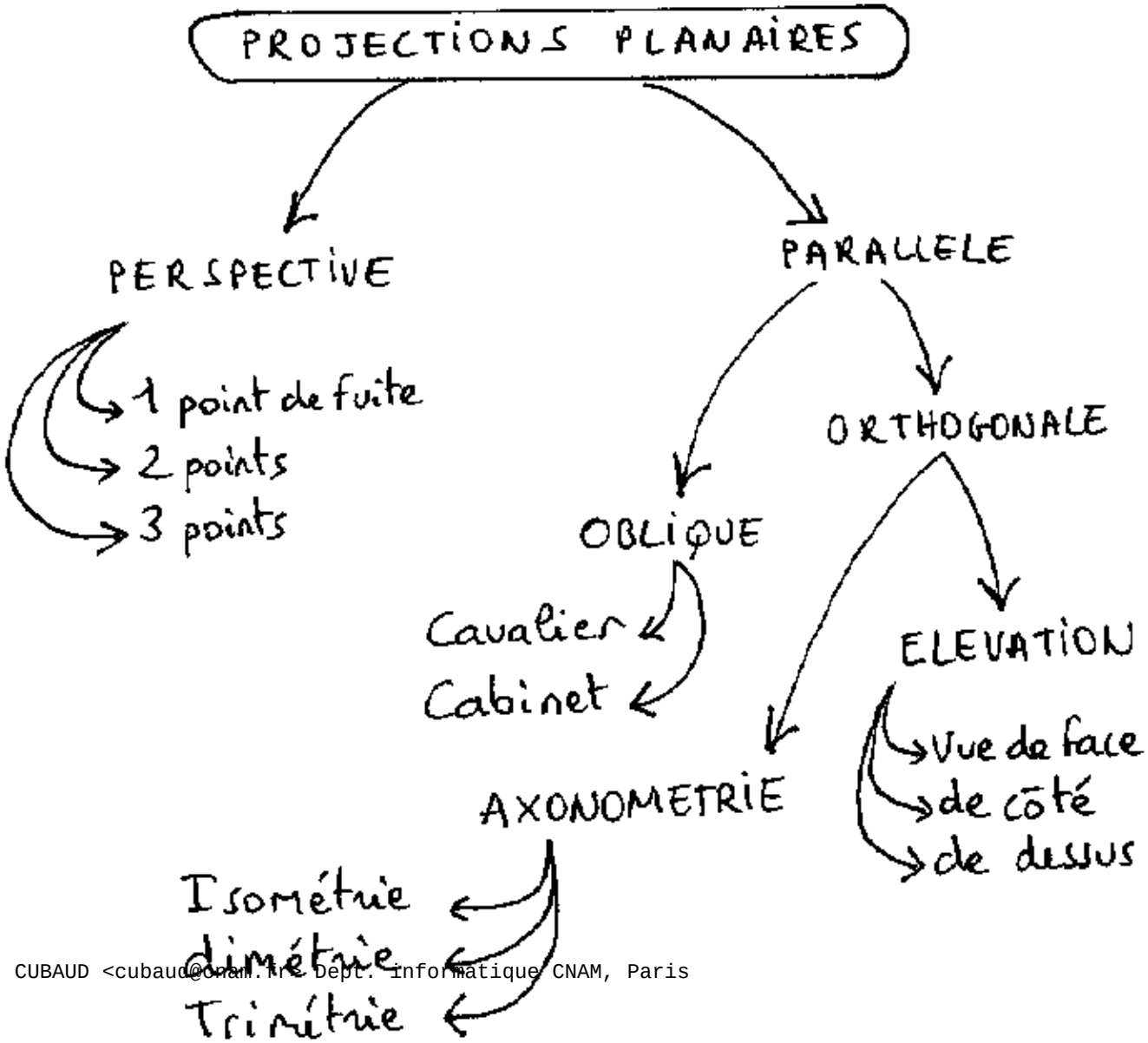
2- Calculs de projection

3- Clôturage (clipping)

4- Digressions

1. Les projections planaires

Une classification :

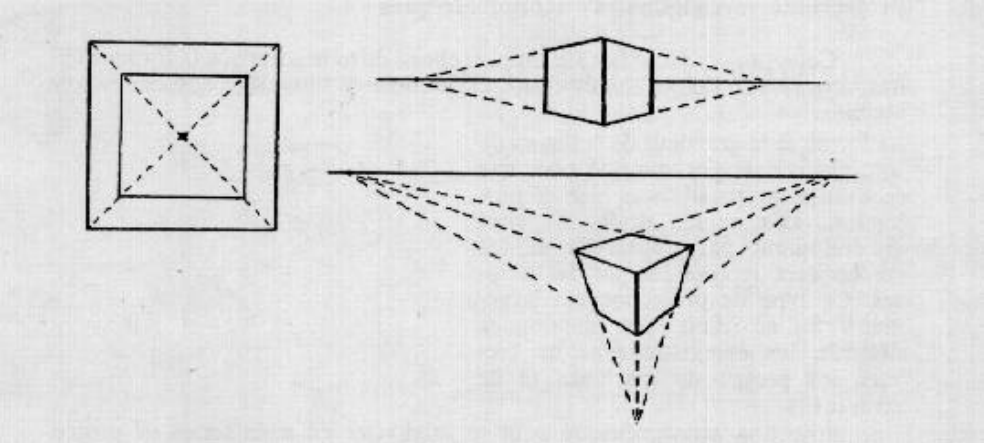


Remarque :

orthogonale = "orthographic" en anglais.

Exemples

- Perspectives



- Cavalier

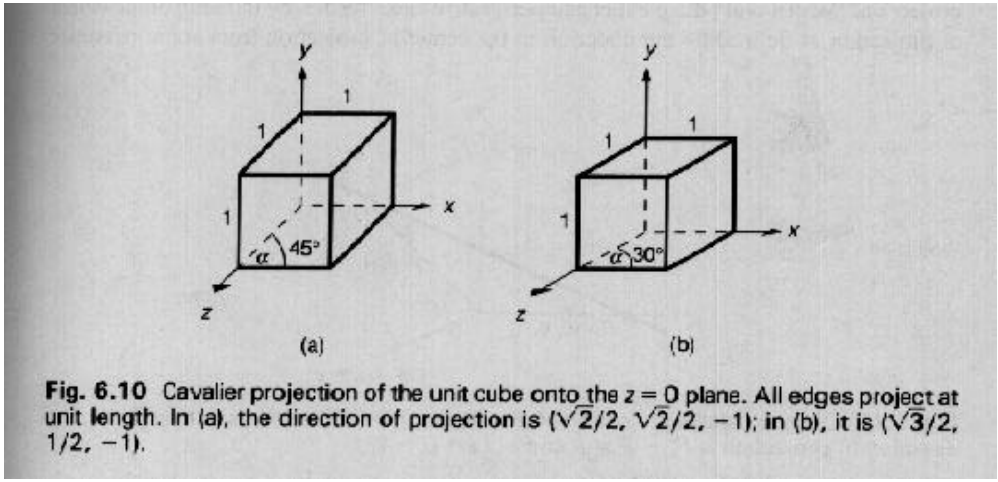


Fig. 6.10 Cavalier projection of the unit cube onto the $z = 0$ plane. All edges project at unit length. In (a), the direction of projection is $(\sqrt{2}/2, \sqrt{2}/2, -1)$; in (b), it is $(\sqrt{3}/2, 1/2, -1)$.

- Cabinet [FOLEY chap. 6]

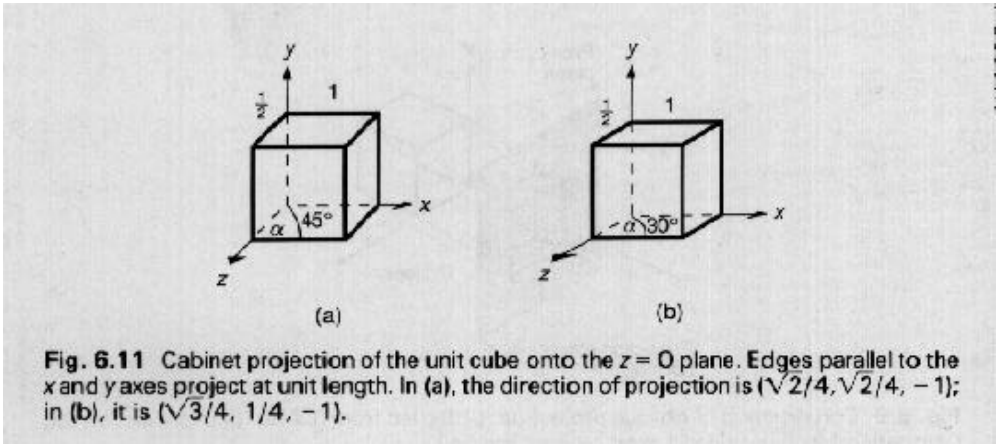


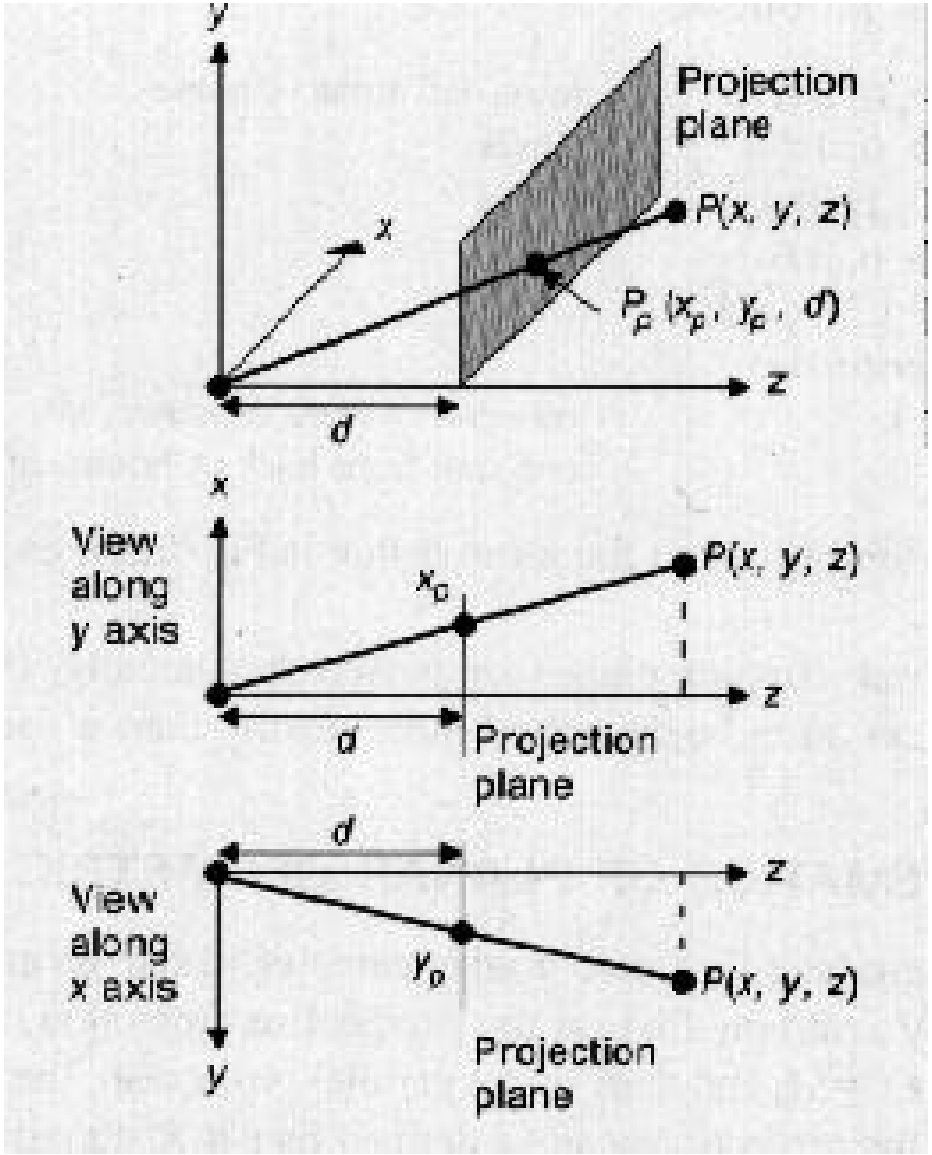
Fig. 6.11 Cabinet projection of the unit cube onto the $z = 0$ plane. Edges parallel to the x and y axes project at unit length. In (a), the direction of projection is $(\sqrt{2}/4, \sqrt{2}/4, -1)$; in (b), it is $(\sqrt{3}/4, 1/4, -1)$.

- Axonométrie

2. Calcul de la projection (I)

Hypothèse simplifiante : le plan de projection est perpendiculaire à l'axe Z

(encore plus simple : plan = XY)



[FOLEY] p. 254

- Projection du pauvre ("poorman 3D")

$$\begin{cases} X = x - \frac{z}{2} \\ Y = y - \frac{z}{2} \end{cases}$$ (repère "main droite")

- Projection parallèle :

On se donne la direction de la projection DDP sous la forme d'un vecteur (xp,yp,zp)

$$\begin{cases} X = x - z\frac{x_P}{z_P} \\ Y = y - z\frac{y_P}{z_P} \end{cases}$$

- Projection perspective :

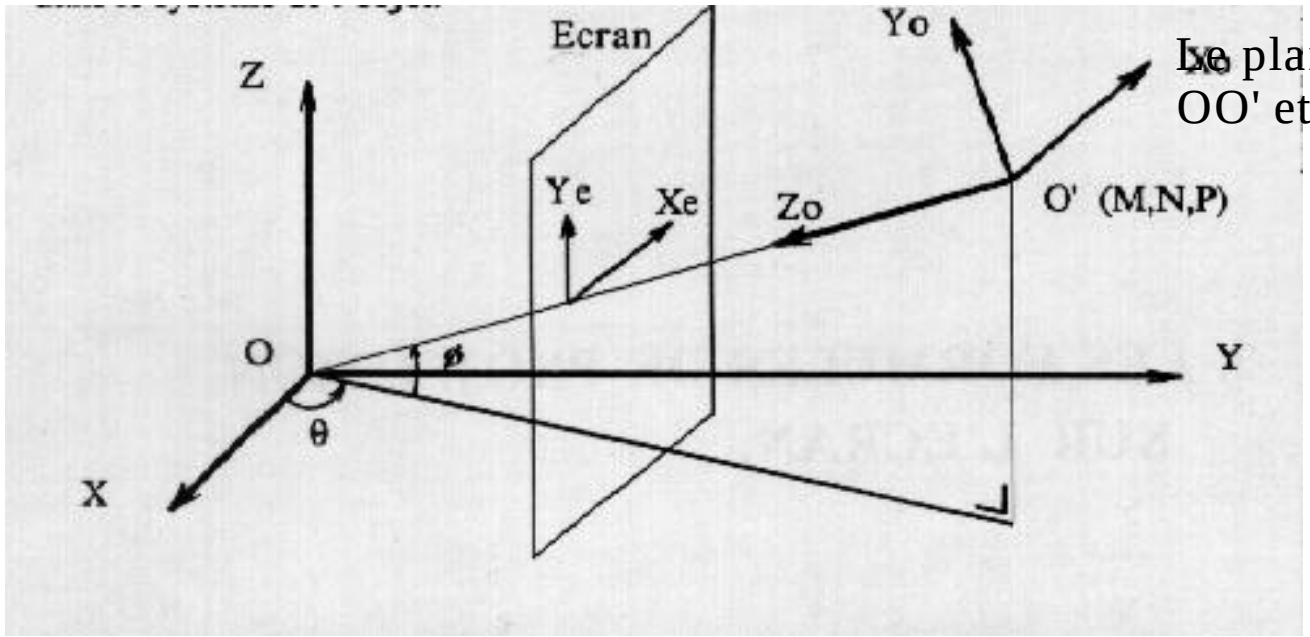
On se donne le point de fuite PDF sous la forme d'un point (xp,yp,zp)

$$\begin{cases} X = \frac{x.z_P - z.x_P}{z_P - z} \\ Y = \frac{y.z_P - z.y_P}{z_P - z} \end{cases}$$

les distances sont "perdues" (les angles aussi)

Calcul de la projection (II)

Généralisation : prise en compte de la position de l'observateur



[DONY] p. 278

La scène est décrite par un repère "main droite" dont l'origine est notée O

L'observateur est en O' et utilise un repère main gauche.

Les coordonnées polaires de O' sont (R, θ, φ)

L'axe Zo du système observateur pointe vers O

Le plan de projection est perpendiculaire à OO' et à une distance D de O'

• 1ère étape : conversion dans le repère de l'observateur

1 - Translation de l'origine O en O' => repère (X1,Y1,Z1)

2 - Rotation du repère (X1,Y1,Z1) de θ-90° autour de Z1 => repère (X2,Y2,Z2)

3 - Rotation du repère (X2,Y2,Z2) de 90°+φ autour de l'axe X2

4 - Conversion en un repère "main gauche"

$$\begin{cases} x' = -x.\sin\theta + y.\cos\theta \\ y' = -x.\cos\theta.\sin\phi - y.\sin\theta.\sin\phi + z.\cos\theta \\ z' = -x.\cos\theta.\cos\phi - y.\sin\theta.\cos\phi - z.\sin\phi + R \end{cases}$$

• 2ème étape : projection du système observé sur l'écran

Si projection perspective :

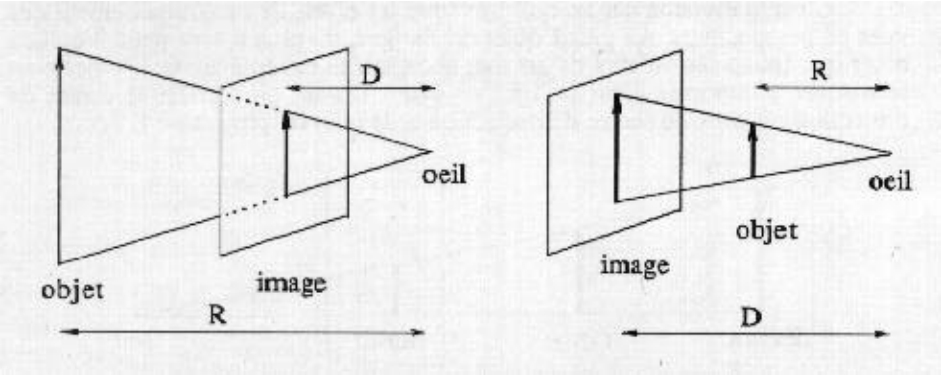
$$X = D\frac{x'}{z'} \qquad Y = D\frac{y'}{z'}$$

Si projection parallèle :

$$X = x' \qquad Y = y'$$

• Remarque : programme complet dans [DONY] pp. 289-294

- Influence des valeurs de R et D



[DONY] p. 284

- Si R augmente, l'image diminue, le centre de projection s'éloigne => ressemble à une projection parallèle

- Si D augmente, la taille de l'image augmente (nulle si D=0 et inversée si plan de projection derrière O')

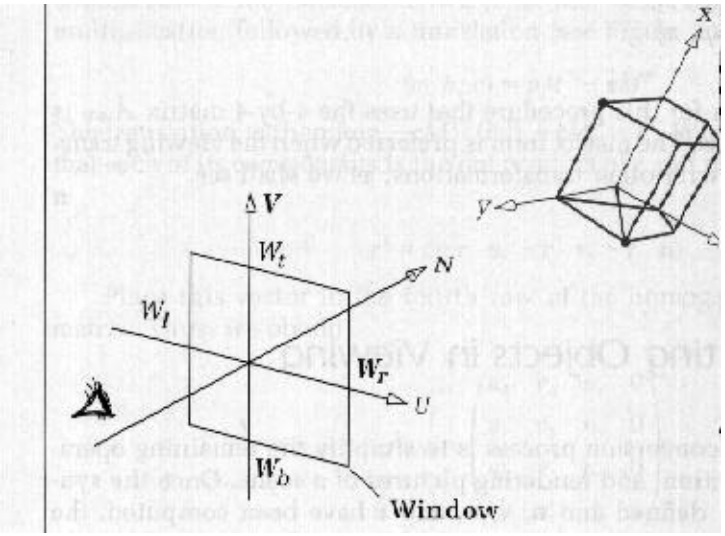
- Si R diminue, alors l'image grandit et se "déforme" si on utilise une projection perspective

- Au lieu d'utiliser un R faible, il vaut mieux prendre un R et un D forts pour agrandir l'image résultante

Calcul de la projection (III)

Généralisation : prise en compte de la direction de l'observation et du champs visuel => caméra synthétique

Méthode utilisée par PHIGS, OpenGL, POV…



[HILL] p. 394

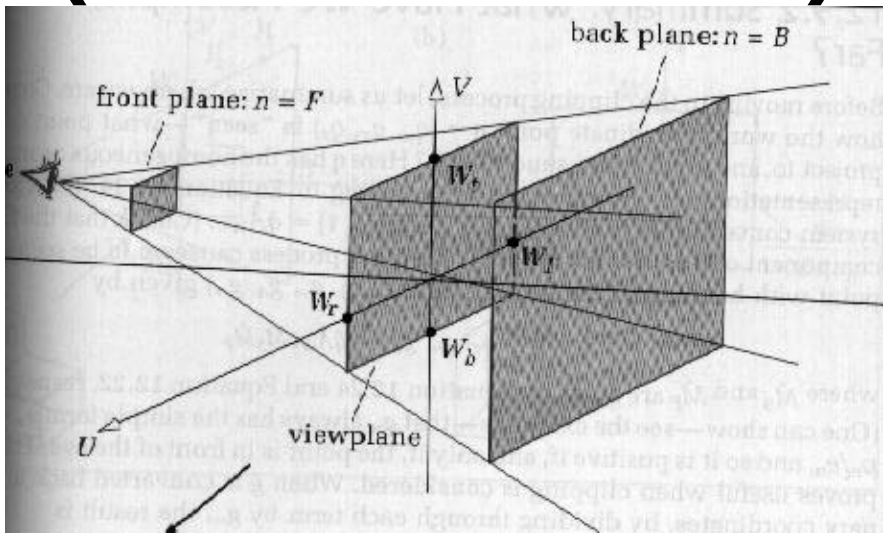
Il faut spécifier :

- la position de l'observateur (comme préced.)
- la direction d'observation, qui sera la normale au plan de projection
- l'orientation de la "tête" par le vecteur V sur le plan de projection
- la taille de l'image projetée sur le plan de projection => une fenêtre

Algorithme complet dans

HILL *Computer graphics* chap. 12

Le volume de vue ("view frustrum")



• En projection parallèle : cube. En projection perspective : pyramide (tronquée)

• Permet d'éviter que les objets situés derrière l'observateur soient eux aussi projetés…

• On peut aussi donner une limite à la profondeur de champ ("back plane")

• On peut aussi "sauter" des élements de la scène situés devant l'observateur en introduisant un "front plane"

3. Le clôturage

• Problème : tous les composants de la scène ne font pas partie du volume de vue.

• Condition suffisante pour être complètement inclus ou exclus du volume de vue : calcul du volume englobant :

1- Calculer la matrice de normalisation pour le volume de vue choisi

2- Normaliser toutes les coordonnées des objets (sommets de polyhèdres ou points de contrôle des surfaces paramétrées)

3 - Rechercher le sommet de plus petite coordonnées

et celui de plus grande

4 - Comparer avec (0,0,0) et (1,1,1)

• Pour les autres : un algorithme qui détermine l'intersection avec le volume de vue.

=> très nombreux algorithmes (cf. [FOLEY] chap 6)

Un exemple : algorithme de Cohen et Sutherland

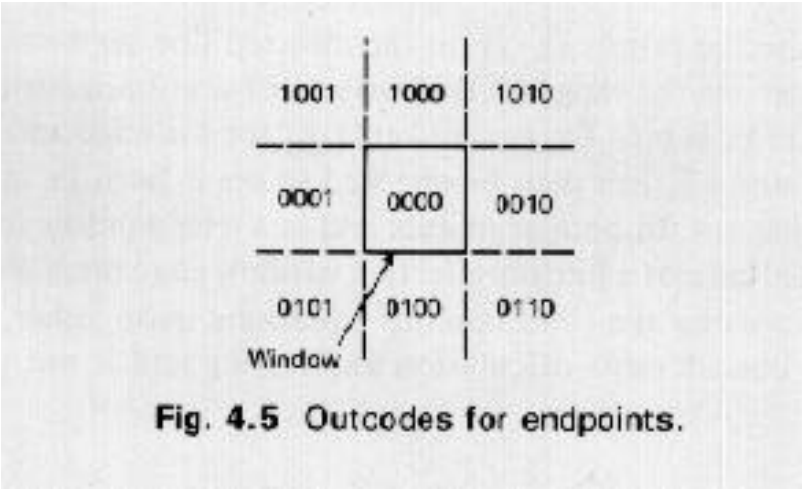
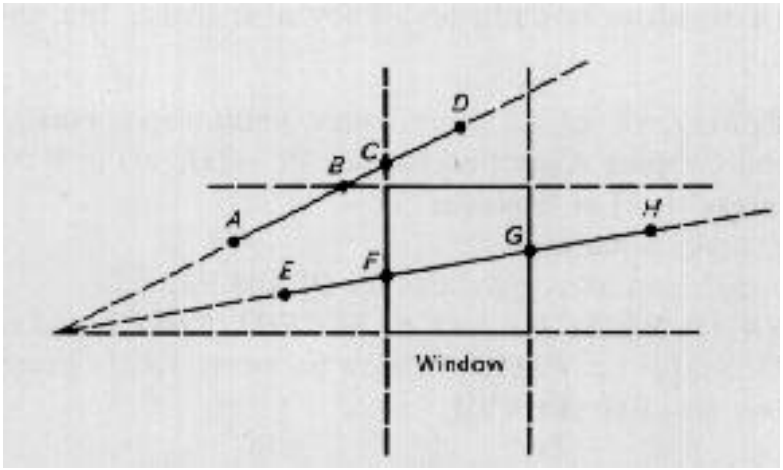


Fig. 4.5 Outcodes for endpoints.

- développé initialement en 2D :



- Encodage de la situation d'un point sur 4 bits

- En 3D : 6 bits pour décrire la position d'un point

• Exemple d'implémentation (proj. parallèle)

```
function clipline3(var p1,p2:vector):boolean;
type sixbits=packed array[1..6] of boolean;
var oc1,oc2,pc:sixbits;p:real;i:integer;

 procedure outcode3(p:vector;var c:sixbits);
 begin
 c[1]:=((1-p[2])<0);c[2]:=(p[2]<0);c[3]:=((1-p[1])<0);
 c[4]:=(p[1]<0);c[5]:=((1-p[3])<0);c[6]:=(p[3]<0)
 end;

 function and6(a1,a2:sixbits):boolean;
 var i:integer;c:boolean;
 begin
 c:=false;
 for i:=1 to 6 do
   if (a1[i] and a2[i]) then c:=true;
 and6:=c
 end;

 function nul6(a:sixbits):boolean;
 begin
 if a[1] or a[2] or a[3] or a[4] or a[5] or a[6] then nul6:=false else
nul6:=true
 end;

begin
 outcode3(p1,oc1);outcode3(p2,oc2);
 while (and6(oc1,oc2)=false)and((nul6(oc1)=false)or(nul6(oc2)=false)) do
   begin
   if nul6(oc1) then begin
                     pc:=oc1;oc1:=oc2;oc2:=pc;
                     for i:=1 to 3 do begin
                                      p:=p1[i];p1[i]:=p2[i];p2[i]:=p
                                      end
                     end;
    if oc1[1] then begin
                   p1[1]:=p1[1]+((p2[1]-p1[1])/(p2[2]-p1[2]))*(1-p1[2]);
                   p1[3]:=p1[3]+((p2[3]-p1[3])/(p2[2]-p1[2]))*(1-p1[2]);
                   p1[2]:=1
                   end else
    if oc1[2] then begin
                   p1[1]:=p1[1]-((p2[1]-p1[1])/(p2[2]-p1[2]))*p1[2];
                   p1[3]:=p1[3]-((p2[3]-p1[3])/(p2[2]-p1[2]))*p1[2];
                   p1[2]:=0
                   end else
    if oc1[3] then begin
                   p1[2]:=p1[2]+((p2[2]-p1[2])/(p2[1]-p1[1]))*(1-p1[1]);
                   p1[3]:=p1[3]+((p2[3]-p1[3])/(p2[1]-p1[1]))*(1-p1[1]);
                   p1[1]:=1
                   end else
    if oc1[4] then begin
                   p1[2]:=p1[2]-((p2[2]-p1[2])/(p2[1]-p1[1]))*p1[1];
                   p1[3]:=p1[3]-((p2[3]-p1[3])/(p2[1]-p1[1]))*p1[1];
                   p1[1]:=0
                   end else
    if oc1[5] then begin
                   p1[1]:=p1[1]+((p2[1]-p1[1])/(p2[3]-p1[3]))*(1-p1[3]);
                   p1[2]:=p1[2]+((p2[2]-p1[2])/(p2[3]-p1[3]))*(1-p1[3]);
                   p1[3]:=1
                   end else
    if oc1[6] then begin
                   p1[1]:=p1[1]-((p2[1]-p1[1])/(p2[3]-p1[3]))*p1[3];
                   p1[2]:=p1[2]-((p2[2]-p1[2])/(p2[3]-p1[3]))*p1[3];
                   p1[3]:=0
                   end
    outcode3(p1,oc1)
 end;(*while*)
 if and6(oc1,oc2)=false then clipline3:=true else clipline3:=false
end;
```

Calcul de la position finale à l'écran

- Il faut indiquer sur quelle portion de l'ecran sera dessinée la fenêtre sur le plan de projection.

- Coordonnées écran :

  - Cas général : le point origine est en haut à gauche de l'écran et en repère main gauche. L'unité est le PIXEL

  - En Postscript : le point origine est (par défaut) en bas à gauche et en repère main droite. L'unité est le PICA

- Exemple de calcul :

  - On choisit deux points de référence dans la scène, en général, une diagonale de la fenêtre de projection : (Xinf, Yinf) et (Xsup, Ysup)

  - On veut que ces points soient situés respectivement en (Xa,Ya) et (Xb,Yb)

  - Pour tous les autres points :

$$\begin{cases} X_{ecran} = \dfrac{X_b - X_a}{X_{\max} - X_{\min}}\left(X - X_{\min}\right) + X_a \\ Y_{ecran} = \dfrac{Y_b - Y_a}{Y_{\max} - Y_{\min}}\left(Y - Y_{\min}\right) + Y_a \end{cases}$$

4. Digressions…

• P. Comar "La perspective en jeu" - déjà cité en introduction.

• (re)lire E. A. Abbott "Flatland"

ftp://ftp.cnam.fr/pub/Gutenberg/etext94/flat10.txt

• Etudier le 4D et les projections vers le 3D

ouvrage chez Belin (titre ???)

• Lire absolument la série de J. Baltrusaitis "Les perspectives dépravées" (Champs Flammarion), consacrée entre autre aux anamorphoses :

• Les perspectives déformantes sont traitées dans [HILL] chap. 11-11

• Vision stéréographique

www.stereoscopy.com

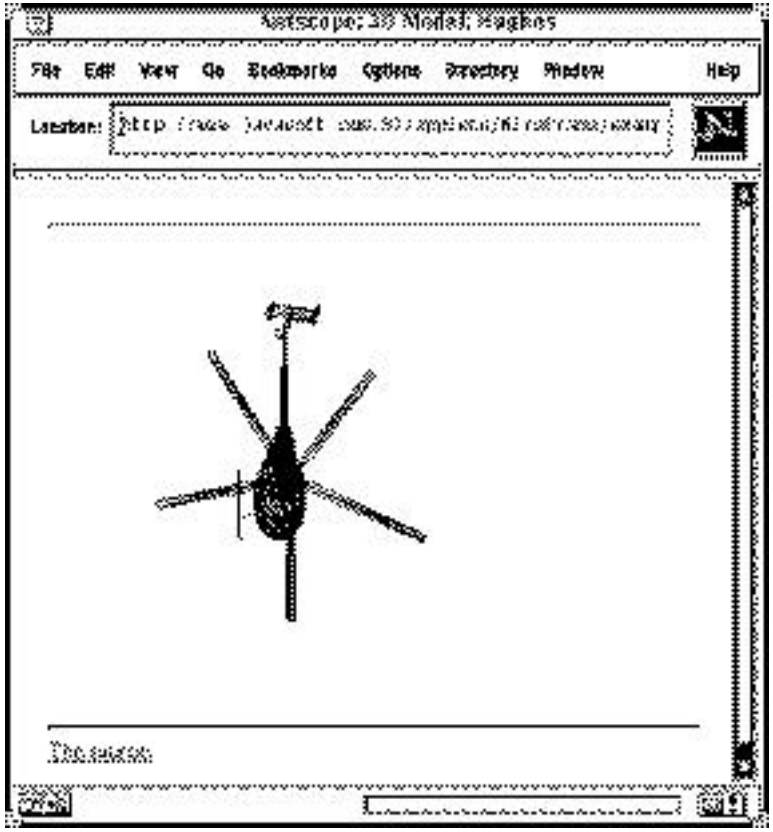
www.stereoskopie.ch

- A étudier : un petit logiciel de visualisation

Une "applet" Java :

http://www.javasoft.com:81/applets/WireFrame/

L'utilisateur choisi avec la souris la position de l'observateur. Si le modèle est simple (ou l'ordinateur puissant), le changement se fait "en temps réel"



Le code est fourni, il est court et très lisible