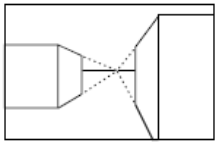
Affichage stéréoscopique

Perception de la profondeur

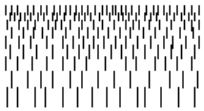
- Un certain nombre d'indices sont utilisés par le système visuel humain pour percevoir la profondeur / la distance
- Ces indices sont souvent présents dans les images en 2D

Perception de la profondeur (2D)



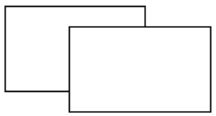
- Perspective: les objets deviennent plus petits lorsqu'ils sont loin et les lignes parallèles convergent.

- Tailles des objets connus: nous nous attendons à une certaine taille des objets. Si un éléphant et une tasse de thé apparaissent de la même taille, on comprend que l'éléphant doit être plus loin que la tasse.

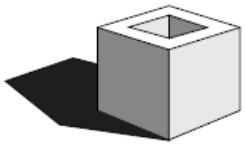


- Niveau de détail: les objets proches apparaissent avec plus de détails que les objets lointains.

Perception de la profondeur (2D)



- Occlusion: un objet qui bloque la vue d'une autre doit être plus près de nous.

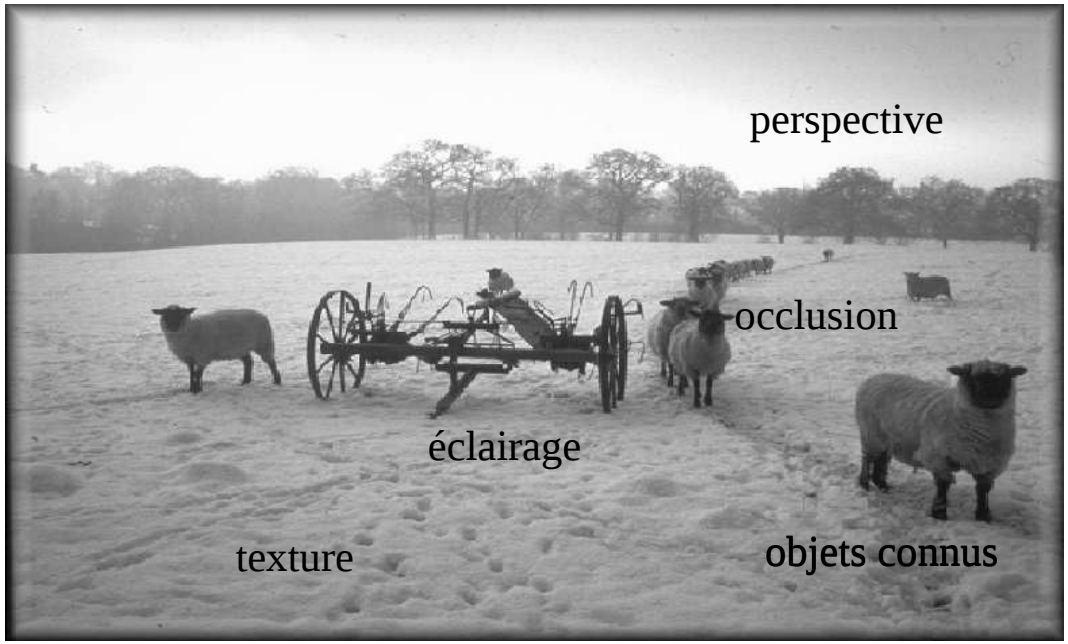


- L'éclairage, les ombres: les objets près de nous sont souvent plus éclairés que les objets lointains.

- Le mouvement relatif: les objets lointains semblent se déplacer plus lentement que les objets au premier plan.

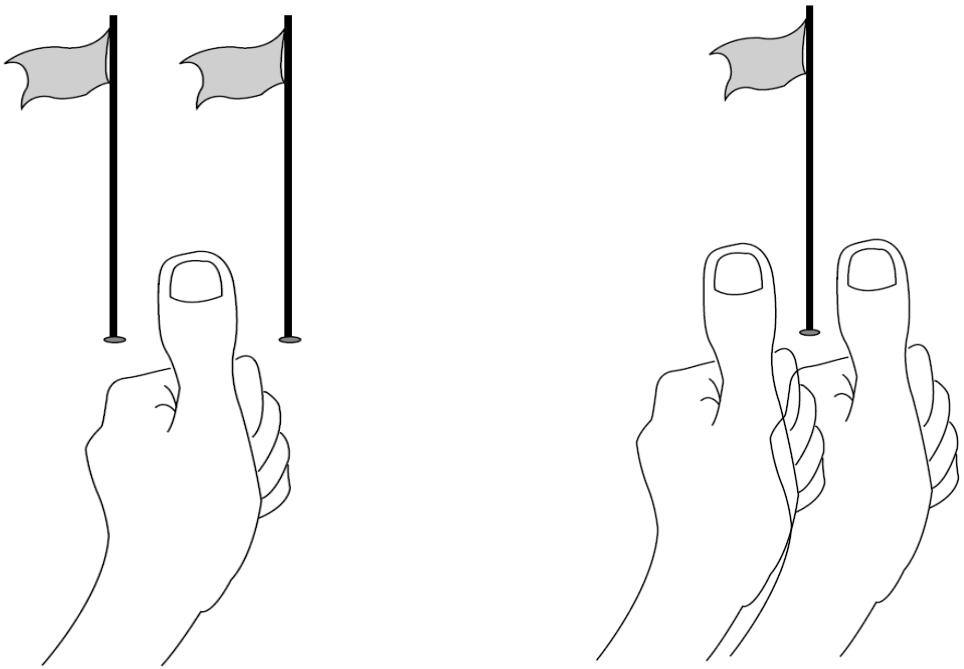
Perception de la profondeur (2D)

Ces indices et d'autres sont souvent présents dans les images



Perception de la profondeur (3D)

D'autres indices sont aussi utilisés par le système visuel humain pour percevoir la profondeur / la distance en 3D.

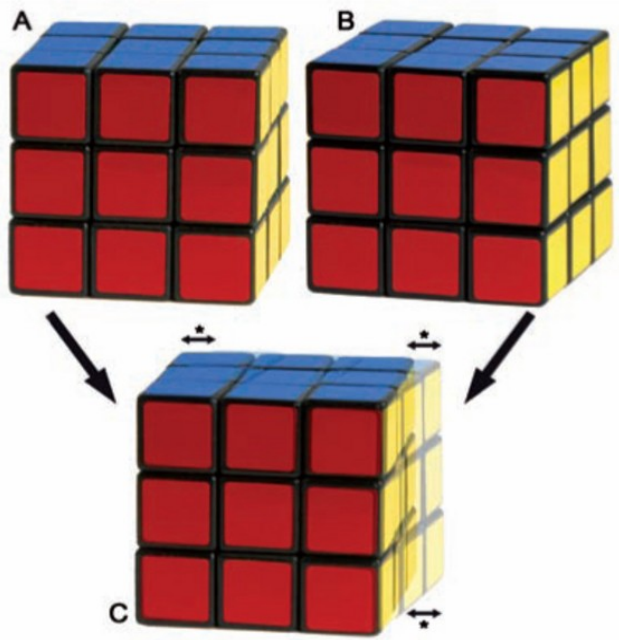


Perception de la profondeur (3D)

D'autres indices ... en 3D :

- Disparité binoculaire : C'est la différence dans les images projetées sur la rétine de l'œil, parce que les yeux sont séparés horizontalement par la distance interoculaire.

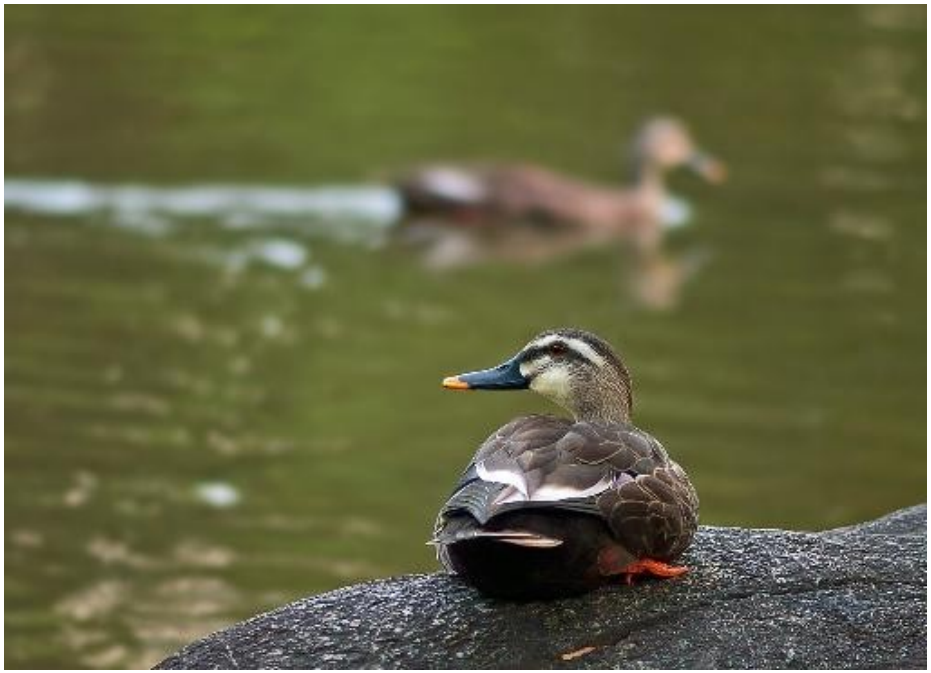
  (65 mm en moyenne)



Perception de la profondeur (3D)

D'autres indices ... en 3D :

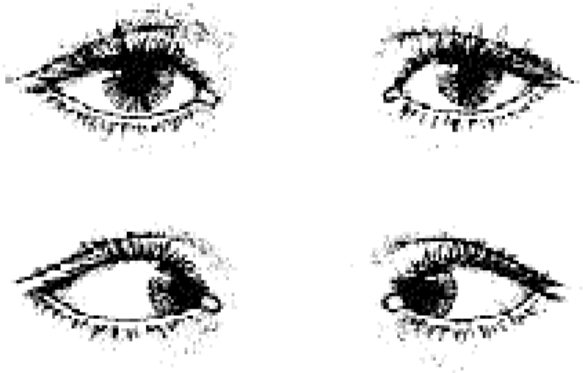
- Distance focale : Il s'agit de la tension musculaire nécessaire pour modifier la distance focale de la lentille de l'œil afin de se concentrer à une profondeur particulière.

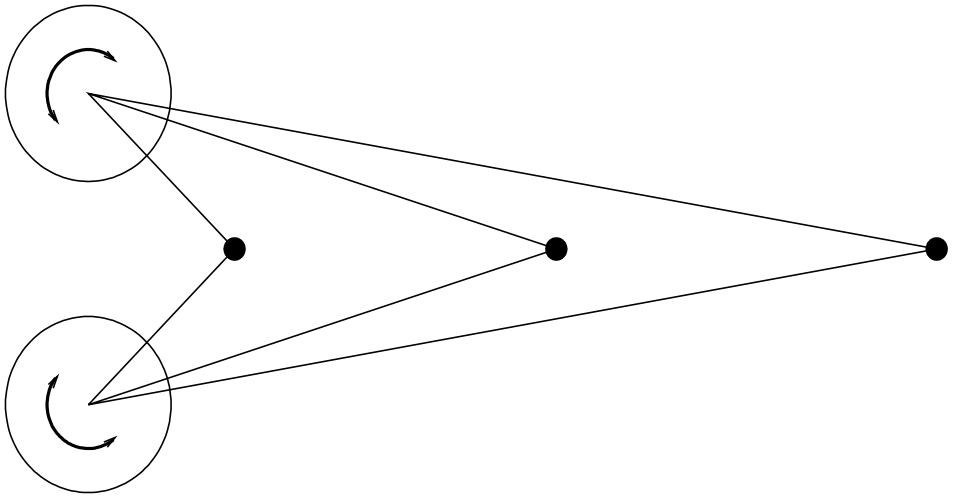


Perception de la profondeur (3D)

D'autres indices ... en 3D :

- Convergence : Il s'agit de la tension musculaire nécessaire pour faire tourner chaque œil afin qu'il soit orienté vers le point focal.





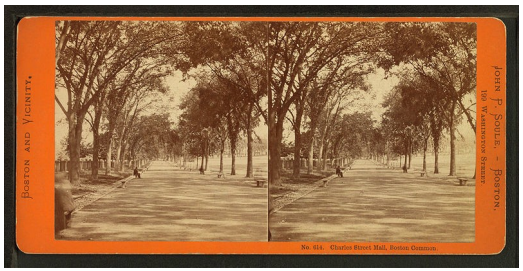
Stéréoscopie (un peu d'histoire)

- Inventé par Sir Charles Wheatstone en 1838
- La technique est de présenter deux images en provenance de deux points de vue différents
- À l'aide d'une technique quelconque, chaque oeil voit une image différente et le cerveau recrée une profondeur

Méthodes pour voir en stéréo

- Sans équipement :
  - L`observateur regarde les deux images simultanément
  - Il faut regarder vers le lointain (vision parallèle)
  - Difficile à faire avec des yeux normaux!

- En croisant les yeux :
  - On échange les images droite et gauche
  - On regarde en croisant les yeux
  - Pas beaucoup plus facile!



Méthodes pour voir en stéréo

- Avec un stéréoscope :
  - Deux lentilles pour envoyer le point focal vers l'infini
  - Ceci permet à l'oeil d'être consistant avec des lignes de vue en parallèle

Stéréoscopes anciens :

Méthodes pour voir en stéréo

Stéréoscopes plus modernes :

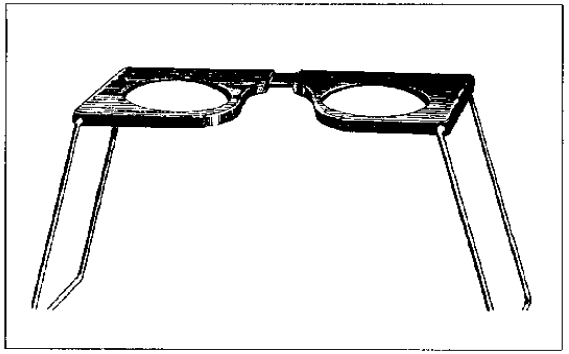


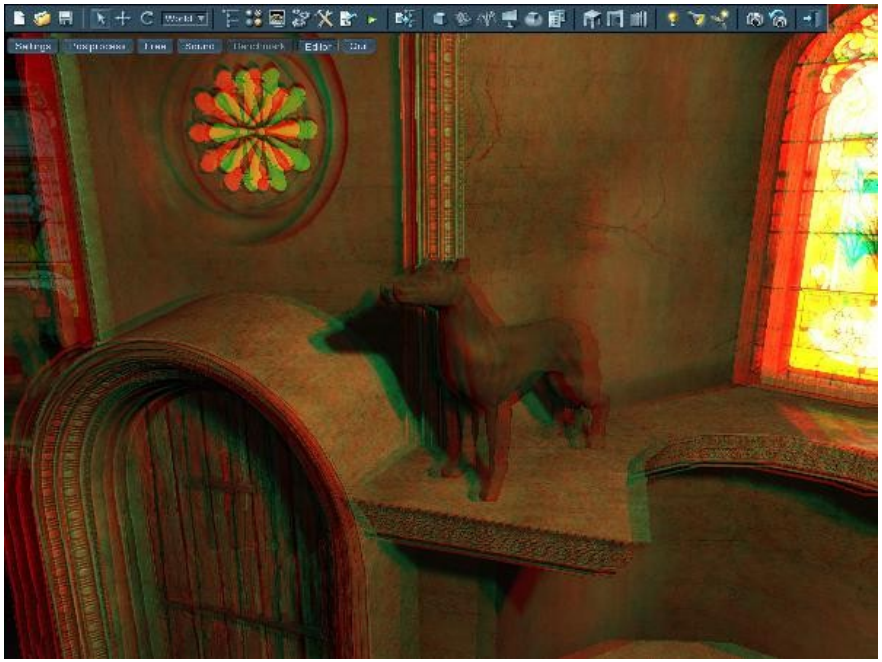
Figure 8-22. Pocket stereoscope.

Méthodes pour voir en stéréo

- En utilisant un anaglyphe :
  - Utilise des couleurs complémentaires pour encoder chaque vue
  - Les filtres les plus courants sont rouge et cyan
    - Le filtre rouge admet le rouge et bloque le vert et bleu
    - Le filtre cyan admet le vert et bleu et bloque le rouge

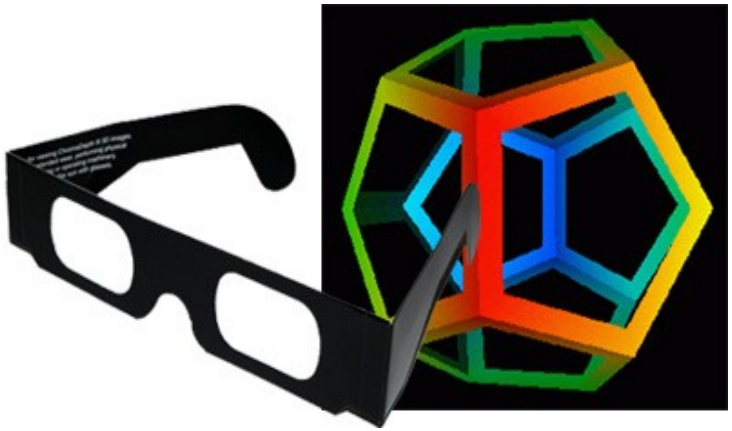
Méthodes pour voir en stéréo

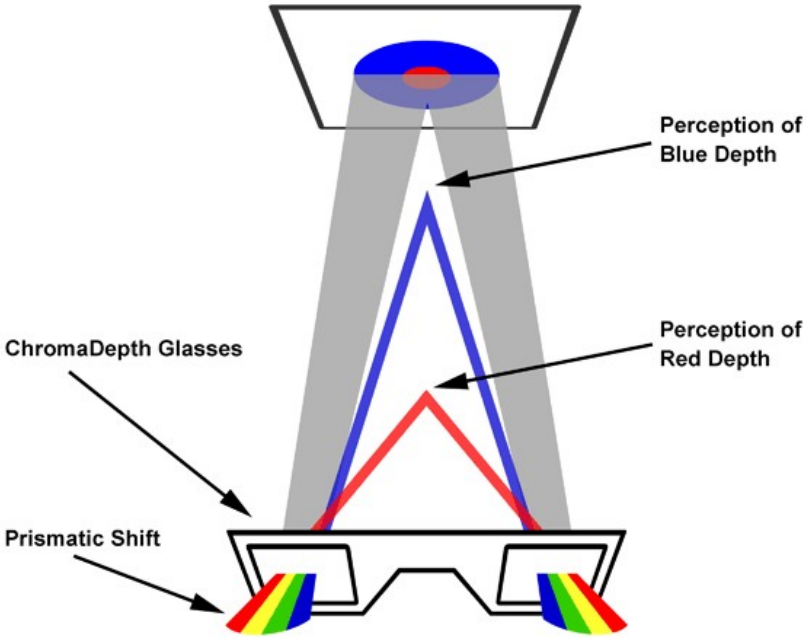
- deux anaglyphes :



Méthodes pour voir en stéréo

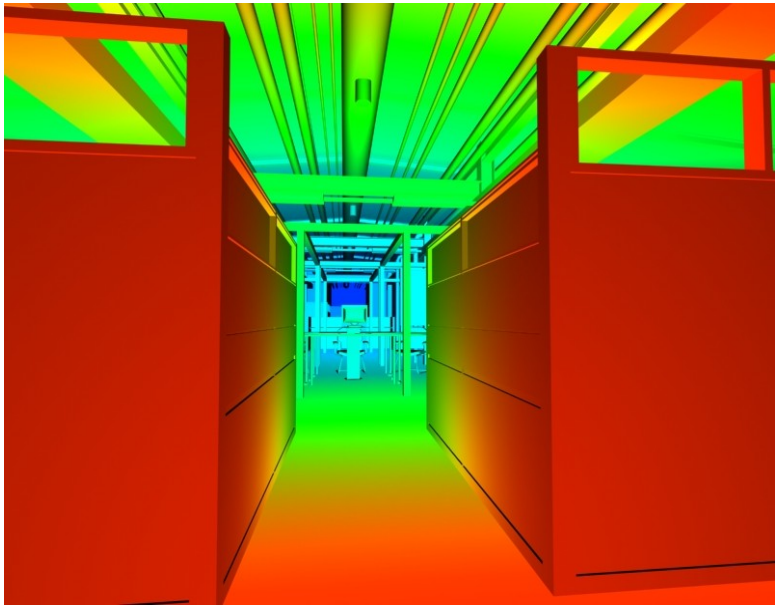
- *Chromadepth*
  - Deux petits prismes séparent différemment les couleurs selon leur longueur d'onde

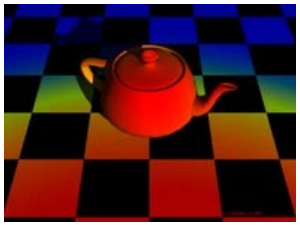




Méthodes pour voir en stéréo

- *Chromadepth*





Méthodes pour voir en stéréo

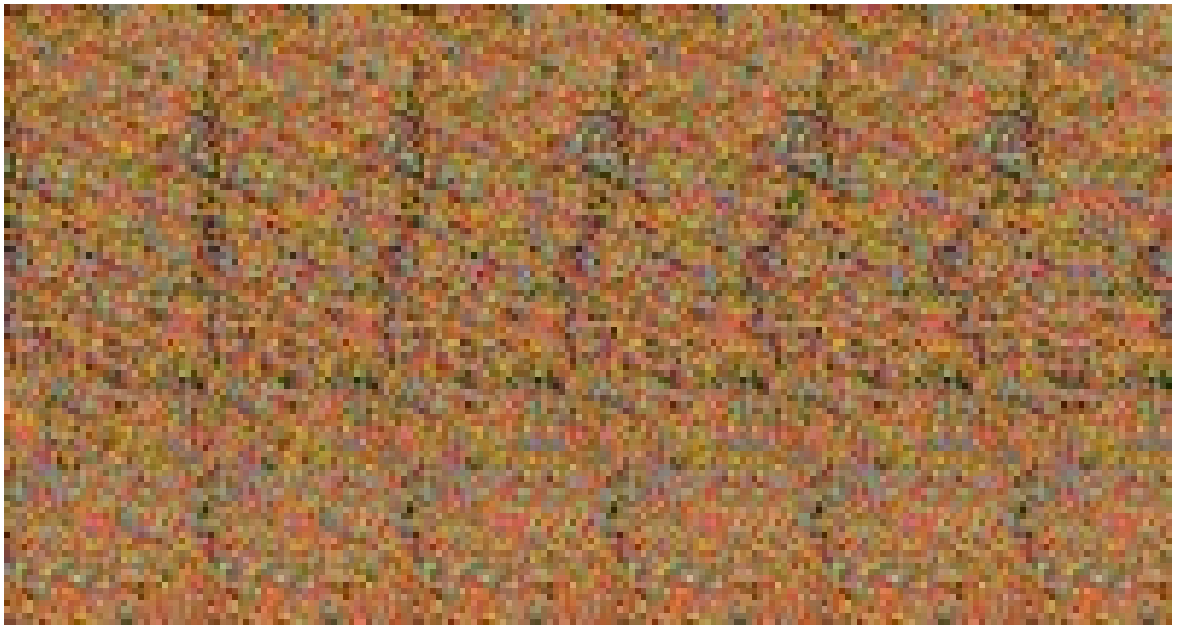
- Avec des lunettes stéréo actives ou passives :
  - Lunettes à cristaux liquides
  - Lunettes polarisées

Méthodes pour voir en stéréo

- Avec un visiocasque :
  - L`utilisateur porte un casque avec deux petits écrans (LCD ou OLED)
  - Souvent couplé avec repérage
  - Utilisé en réalité virtuelle ou en réalité augmentée

Méthodes pour voir en stéréo

- Auto-stéréogramme
  - Des points de couleur cachent une image qu'on peut voir lorsque les yeux convergent ou divergent



Méthodes pour voir en stéréo

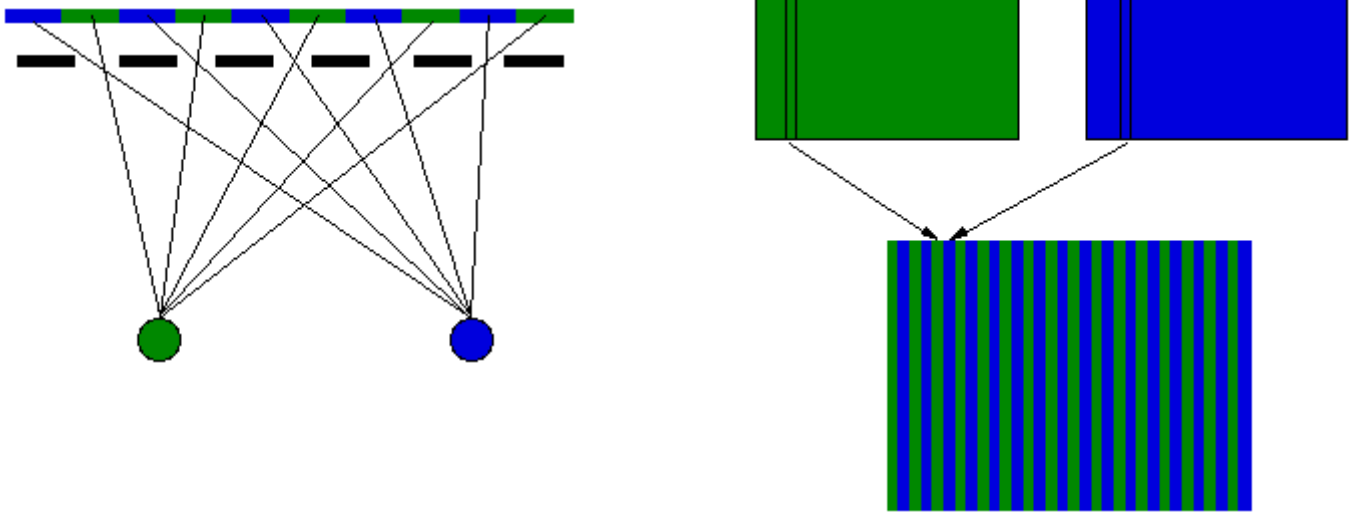
- Écran auto-stéréoscopique
  - Écran qui n'ont pas besoin de lunettes pour voir du 3D

(pour illustration seulement : ce n'est pas possible ainsi!)

  - Deux techniques pour produire les images :
    - Avec une barrière de parallaxe
    - Avec une barrière lenticulaire

Méthodes pour voir en stéréo

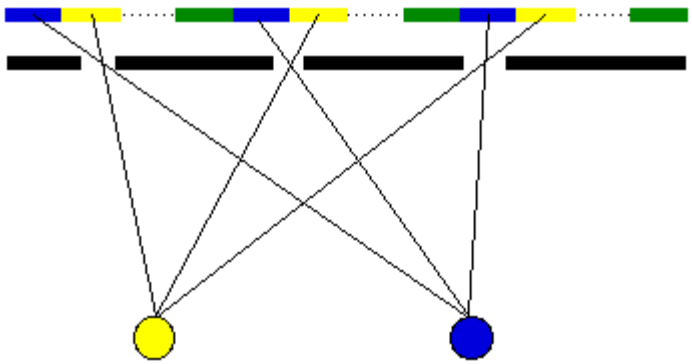
- Écran auto-stéréoscopique
  - En utilisant une barrière de parallaxe



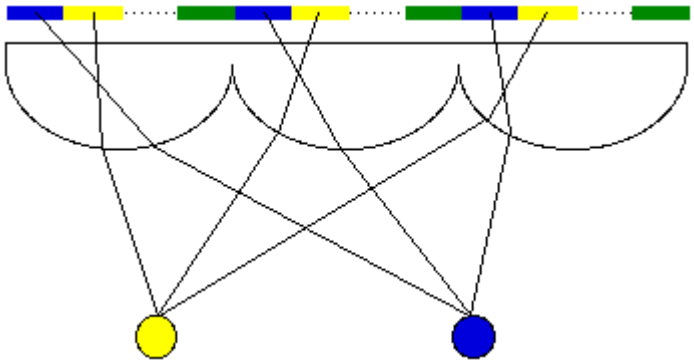
Méthodes pour voir en stéréo

- Écran auto-stéréoscopique
  - En utilisant une barrière lenticulaire

parallaxe

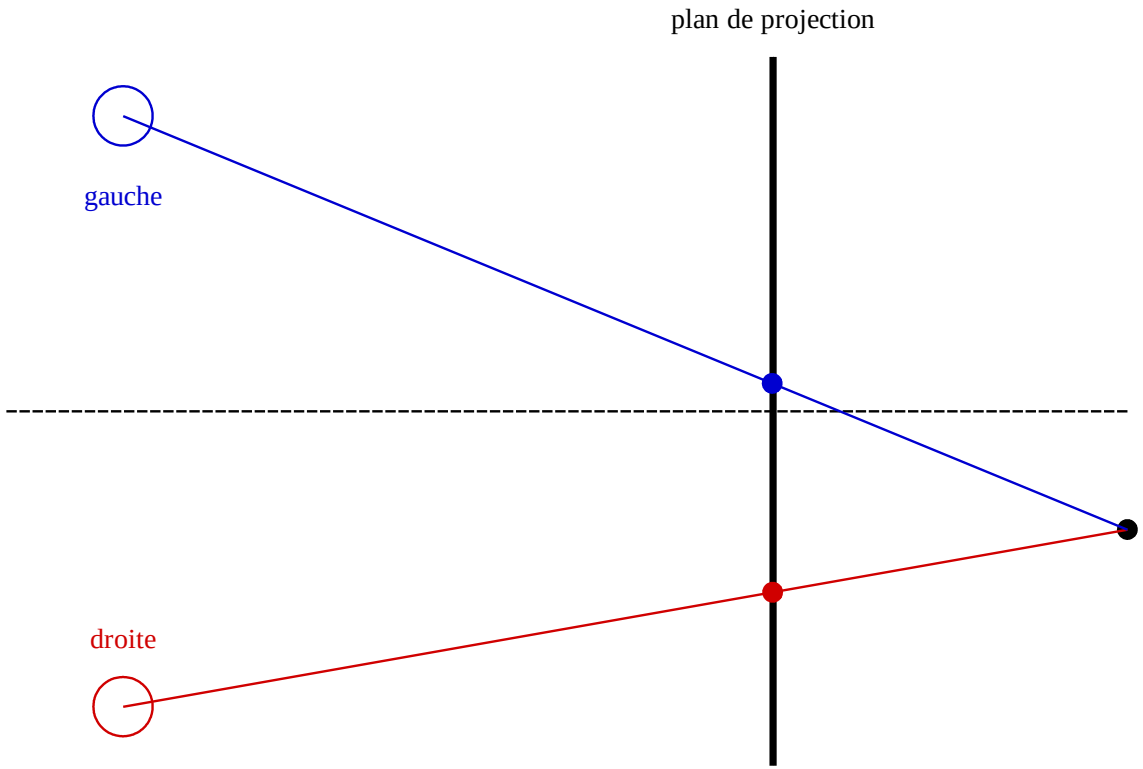


lenticulaire



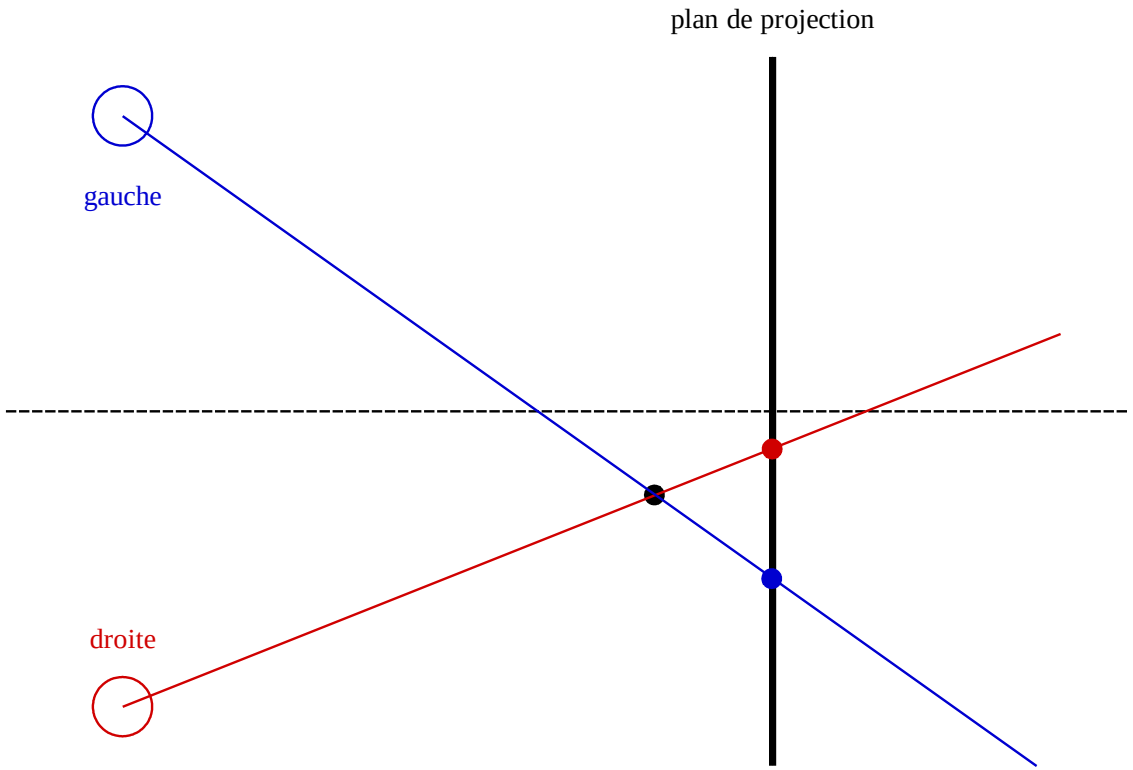
Fonctionnement avec OpenGL

- Parallaxe positive : le point projeté est en arrière de l'écran.



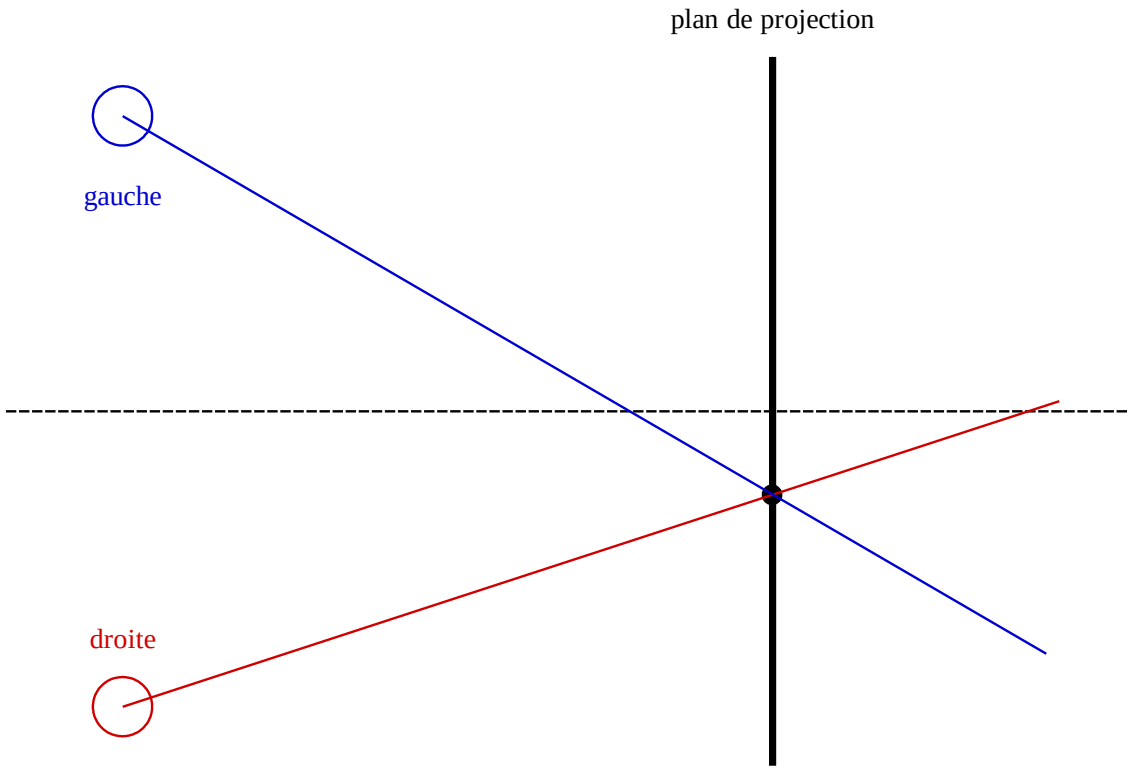
Fonctionnement avec OpenGL

- Parallaxe négative : le point projeté est en avant de l'écran.



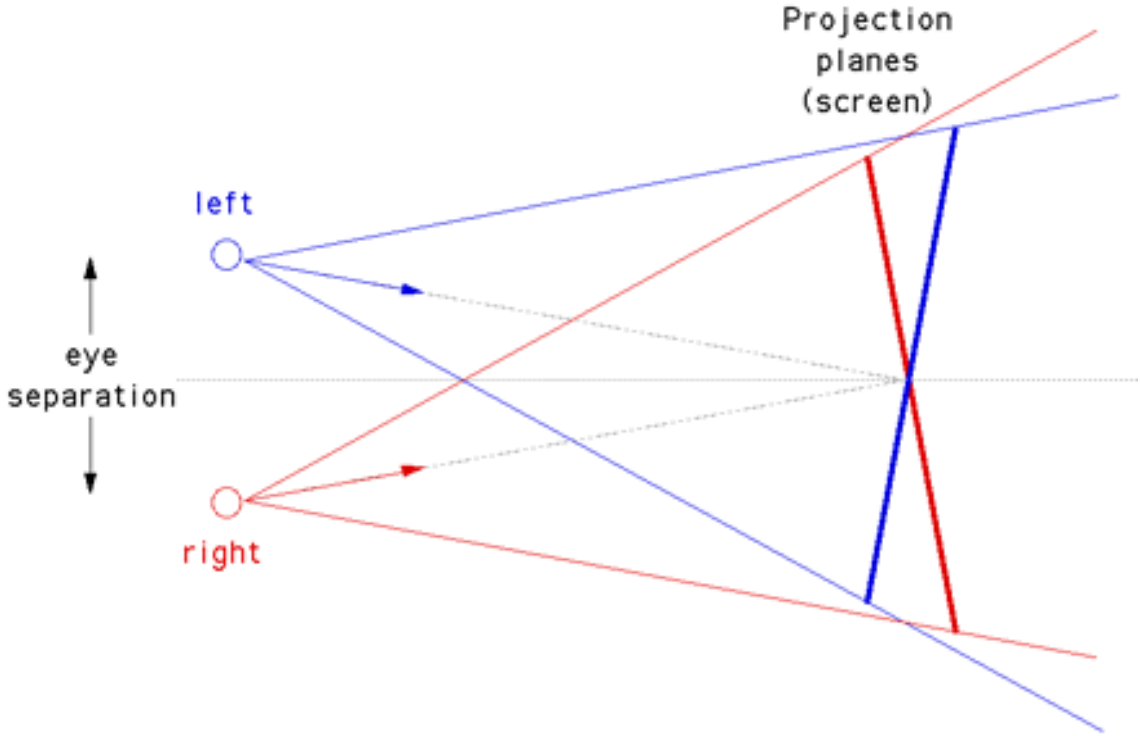
Fonctionnement avec OpenGL

- Parallaxe zéro : le point projeté est sur l'écran.



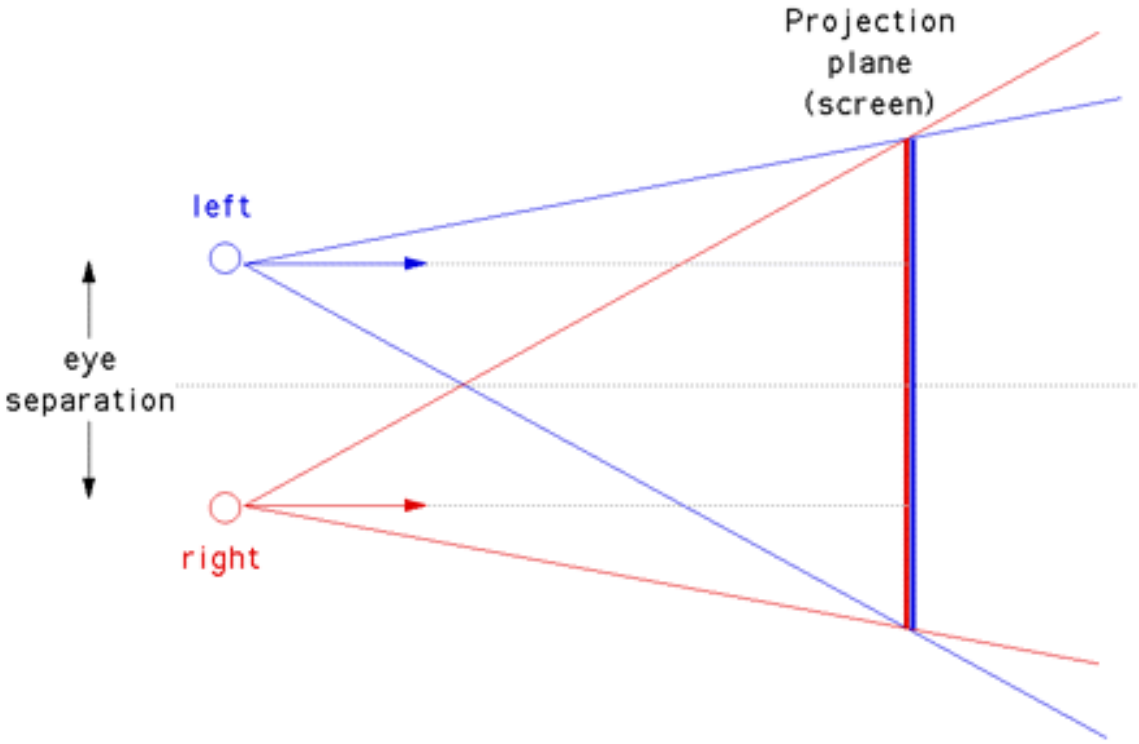
Fonctionnement avec OpenGL

- Méthode de rendu facile, mais *incorrecte*!



Fonctionnement avec OpenGL

- Méthode de rendu correcte



Fonctionnement avec OpenGL

- Méthode de rendu facile, mais *incorrecte*!

```
glMatrixMode( GL_PROJECTION );
glLoadIdentity( ); gluPerspective( ... );
glMatrixMode( GL_MODELVIEW );
if ( !stereo )
{
   glDrawBuffer( GL_BACK );
   glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );
   glLoadIdentity( );
   gluLookAt( cam.pos.x, cam.pos.y, cam.pos.z,
              cam.ptvise.x, cam.ptvise.y, cam.ptvise.z,
              cam.up.x, cam.up.y, cam.up.z );
   afficher( );
}
else
```

Fonctionnement avec OpenGL

```
{
   PRODSCAL( cam.ptvise, cam.up, depl ); Normalise( &depl );
   depl.x *= cam.eyesep/2.0; depl.y *= cam.eyesep/2.0; depl.z *= cam.eyesep/2.0;
   // image droite
   glDrawBuffer( GL_BACK_RIGHT );
   glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );
   glLoadIdentity( );
   gluLookAt( cam.pos.x + depl.x, cam.pos.y + depl.y, cam.pos.z + depl.z,
              cam.ptvise.x, cam.ptvise.y, cam.ptvise.z,
              cam.up.x, cam.up.y, cam.up.z );
   afficher( );
   // image gauche
   glDrawBuffer( GL_BACK_LEFT );
   glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );
   glLoadIdentity( );
   gluLookAt( cam.pos.x – depl.x, cam.pos.y – depl.y, cam.pos.z - depl.z,
              cam.ptvise.x, cam.ptvise.y, cam.ptvise.z,
              cam.up.x, cam.up.y, cam.up.z );
   afficher( );
}
```

Fonctionnement avec OpenGL

- Méthode de rendu correct

```
ratio   = cam.screenwidth / ( double )cam.screenheight;
radians = DTOR * cam.aperture / 2;
wd2     = near * tan( radians );
ndfl    = near / cam.focallength;
// ...
```

Fonctionnement avec OpenGL

```
if ( !stereo )
{
   glMatrixMode( GL_PROJECTION );
   glLoadIdentity( );
   left  = - ratio * wd2;
   right =   ratio * wd2;
   top    =   wd2;
   bottom = - wd2;
   glFrustum( left, right, bottom, top, near, far );
   glDrawBuffer( GL_BACK );
   glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );
   glMatrixMode( GL_MODELVIEW );
   glLoadIdentity( );
   gluLookAt( cam.pos.x, cam.pos.y, cam.pos.z,
              cam.ptvise.x, cam.ptvise.y, cam.ptvise.z,
              cam.up.x, cam.up.y, cam.up.z );
   afficher( );
}
```

Fonctionnement avec OpenGL

```
else
{
   PRODSCAL( cam.ptvise, cam.up, depl ); Normalise( &depl );
   depl.x *= cam.eyesep / 2.0;
   depl.y *= cam.eyesep / 2.0;
   depl.z *= cam.eyesep / 2.0;
```

Fonctionnement avec OpenGL

```
glMatrixMode( GL_PROJECTION );
glLoadIdentity( );
left  = - ratio * wd2 - 0.5 * cam.eyesep * ndfl;
right =   ratio * wd2 - 0.5 * cam.eyesep * ndfl;
top    =   wd2;
bottom = - wd2;
glFrustum( left, right, bottom, top, near, far );

glDrawBuffer( GL_BACK_RIGHT );
glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );
glMatrixMode( GL_MODELVIEW );
glLoadIdentity( );
gluLookAt( cam.pos.x + depl.x, cam.pos.y + depl.y, cam.pos.z + depl.z,
           cam.ptvise.x + depl.x,
           cam.ptvise.y + depl.y,
           cam.ptvise.z + depl.z,
           cam.up.x, cam.up.y, cam.up.z );
afficher( );
```

Fonctionnement avec OpenGL

```
    glMatrixMode( GL_PROJECTION );
    glLoadIdentity( );
    left  = - ratio * wd2 + 0.5 * cam.eyesep * ndfl;
    right =   ratio * wd2 + 0.5 * cam.eyesep * ndfl;
    top    =   wd2;
    bottom = - wd2;
    glFrustum( left, right, bottom, top, near, far );

    glDrawBuffer( GL_BACK_LEFT );
    glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );
    glMatrixMode( GL_MODELVIEW );
    glLoadIdentity( );
    gluLookAt( cam.pos.x - depl.x, cam.pos.y - depl.y, cam.pos.z - depl.z,
               cam.ptvise.x - depl.x,
               cam.ptvise.y - depl.y,
               cam.ptvise.z - depl.z,
               cam.up.x, cam.up.y, cam.up.z );
    afficher( );
}
```