



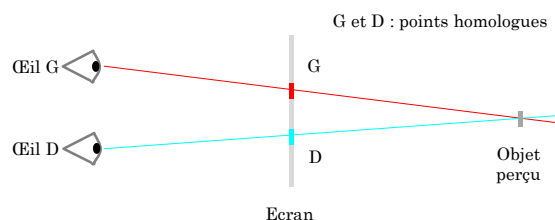
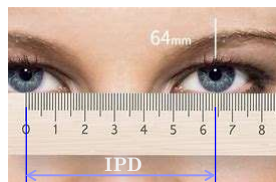
IX – Vision stéréoscopique

127

127

IX.1 – La stéréoscopie

- Objectifs :
 - Visualiser le relief des scènes virtuelles (donc, des objets 3D)
- Principe de perception du relief
 - Principe physiologique !
 - Le cerveau utilise la **parallaxe** de vision
 - 2 images issues de 2 points de vue légèrement espacés
 - « IPD » : « Inter Pupillary Distance » $\approx 65\text{mm}$



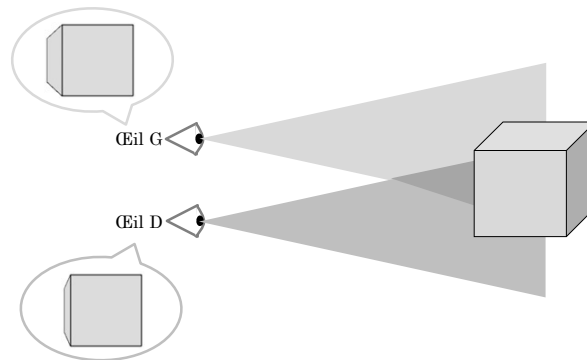
128

128

IX.1 – La stéréoscopie

■ Principe de restitution numérique du relief

- Utiliser / Créer 2 images prises depuis 2 points de vue différents
 - « base stéréoscopique » B : écart des points de vue
- Faire en sorte que chaque œil ne voit que l'image qui le concerne
- **Stéréopsie** :
 - le cerveau « fusionne » les images,
 - en calculant la profondeur, la distance des objets
 - image mentale « unique » en vision binoculaire (avec perspective)



129

129

IX.1 – La stéréoscopie et la géométrie

■ Performance de l'œil humain

- acuité de l'œil : pouvoir de résolution

$$a = 1' \text{ arc} = \frac{1}{60^\circ} = 0,017^\circ = \frac{\pi}{180 \times 60} \approx 3.10^{-4} \text{ rad}$$

■ Géométrie



- Distance minimum discernable sur l'arrière plan : s



$$s \approx a \cdot (D + L)$$

$$e > s$$

- Base stéréoscopique B

$$\text{comme } e > s \text{ alors } \frac{e}{(D + L)} > a$$

$$\text{Thalès : } \frac{B}{D} = \frac{e}{L} \quad \text{donc : } B > \frac{a \cdot D \cdot (D + L)}{L}$$

<http://becot.info/stereo/francais/3Dtheorie.htm>

130

130

- facteur caractérisant la limite de l'écartement



132

3

IX.2 – Relief : parallaxe négative et positive

- Cas extrêmes
 - Vision sans relief
- Divergence interdite

Œil G

Œil D

G

D

Ecran


Œil G

Œil D

G

D

Ecran




133

IX.3 – Restitution stéréoscopique

- Stéréoscopie en vision libre
 - Deux images côte à côte, en vision **croisée**

© 1997 Boris Starosta

© 1997 Boris Starosta




<http://starosta.com/3dshowcase/ihelp.html>

Help: How to Free-View the Stereo Pairs

Two images, one for each eye. Free viewing is the technique that will allow you to direct each of these images separately and simultaneously.

Two pairs of images in loading with which you can practice. All the stereo pairs shown on this site are in the "cross-eyed" format (my apologies to the original author). The left image of each pair is for your left eye and the right image is for your right eye. Looking at the pair in a natural, relaxed manner, what you see will be similar to this:



I am indicating with color the region to which you must direct your attention - that area of overlap in where you will fuse the left and right images of the pair. Fusion (yet)

3D Showcase: Help with Cross-Eyed Stereo Viewing (starosta.com)

134

Emmanuel MESNARD

4

IX.3 – Restitution stéréoscopique vision libre

■ Stéréoscopie en vision libre

- Attention : vision croisée \neq vision parallèle
 - Permutation images gauche / droite
- Avantages
 - Images couleurs
 - Peu coûteux !
- Inconvénients
 - Pas toujours évident de loucher volontairement : aide par point homologue temporaire
 - Petites images, pour faciliter la stéréopsie
 - Chaque œil ne voit pas **exclusivement** son image dédiée : deux « parasites »



[Stereotron-3D - Hobbyist, Photographer | DeviantArt](https://www.deviantart.com/stereotron-3d/gallery)
<https://www.deviantart.com/stereotron-3d/gallery>

135

135

IX.3 – Restitution stéréoscopique passive

■ Stéréoscopie passive

- Filtrage par verres polarisés H/V
 - A l'émission : devant les vidéoprojecteurs
 - A la réception : paire de lunettes devant les yeux



- Avantages
 - Images couleurs
 - Très coûteux : deux vidéoprojecteurs, avec filtres par contre, lunettes très bon marché



- Inconvénients
 - Maintenance délicate : ajustement des projections
 - Deux images superposées, donc, perte de luminosité
 - Filtres imparfaits : « Ghost », **diaphotie** effet fantôme où un œil perçoit l'image destinée à l'autre œil



136

136

IX.3 – Restitution stéréoscopique active

■ Stéréoscopie active

□ Lunettes actives à obturation synchronisée

- Obturation (LCD) à au moins 60 images/s
- Vidéoprojecteur(s) ou moniteur 3D
- Synchronisation : Infrarouge, radio fréquence



□ Avantages

- Images couleurs
- Coûteux : vidéoprojecteurs rapides + lunettes

□ Inconvénients

- Batterie et autonomie des lunettes
- Pertes de synchronisation !
- Occultation partielle des LCD : effet fantôme



137

137

IX.3 – Restitution stéréoscopique anaglyphe

■ Stéréoscopie anaglyphe

□ Filtrage par couleurs complémentaires Rouge / Cyan

- A l'émission : combinaison numérique des images
- A la réception : paire de lunettes anaglyphes
 - Rouge : œil gauche; Cyan œil droit



Rouge			Cyan		
H	0	°	H	179	°
S	99	%	S	99	%
B	99	%	B	99	%
R	255		R	0	
G	0		G	255	
B	0		B	255	

Vert			Bleu		
H	119	°	H	239	°
S	99	%	S	99	%
B	99	%	B	99	%
R	0		R	0	
G	255		G	0	
B	0		B	255	

Cyan = Vert + Bleu

□ Avantages

- Très peu coûteux
- Très bon résultat avec images N&B

□ Inconvénients

- Résultat moins bon en couleurs
- Filtres imparfaits : « Ghost », diaphotie
- Problème de **Rivalité** de couleur
Si couleurs primaires utilisées dans l'image car vues que d'un œil (stéréopsie impossible : un œil est ébloui et pas l'autre : objet instable, vibrant)



138

138

IX.3 – Création d'image stéréoscopique anaglyphe

■ Principe extrêmement simple :

- RGB de l'image finale est constitué
 - de la composante R de l'image de gauche
 - des composantes GB de l'image de droite



□ Matrice de transformation M

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix}_{Image\ Anaglyphe} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} R \\ G \\ B \end{pmatrix}_{Image\ Gauche} \times \begin{pmatrix} R \\ G \\ B \end{pmatrix}_{Image\ Droite}$$



Image Anaglyphe



Image Gauche



Image Droite

139

139

IX.4 – Exemple 16 : Image Anaglyphe

```
...
void setup() {
    size(500, 500);
    // Ouverture et redimensionnement des images
    photo_G = loadImage("photo_G.jpg");    photo_D = loadImage("photo_D.jpg");
    photo_G.resize(500, 500);               photo_D.resize(500, 500);
    Anaglyphe = createImage(500, 500, RGB);

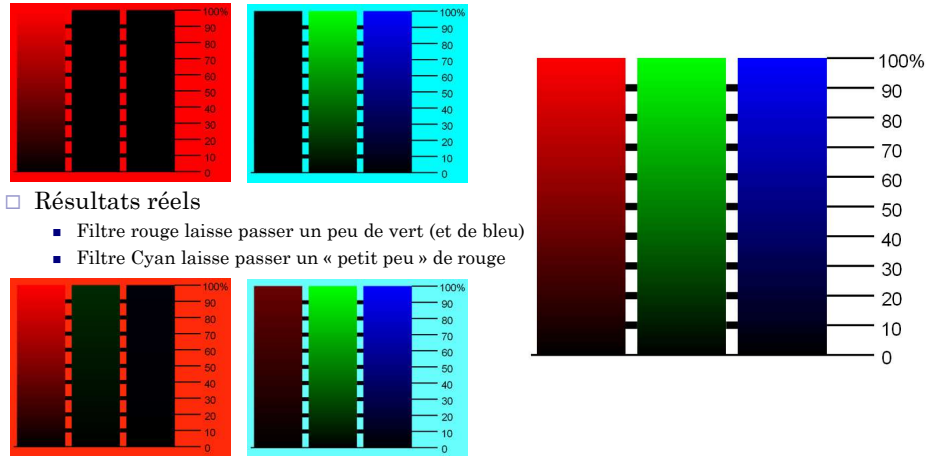
    photo_D.loadPixels();    photo_G.loadPixels();    Anaglyphe.loadPixels();
    for (y = 0; y < 500; y++) { // lignes y
        yDebutLigne = y * 500;
        for (x = 0; x < 500; x++) { // colonnes x
            i = yDebutLigne + x;
            pixelImgG = photo_G.pixels[i];
            pixelImgD = photo_D.pixels[i];
            // Extractions des 3 composantes couleur utiles ici
            Rg = (pixelImgG >> 16) & 0xFF; // ou red(pixelImgG)
            Gd = (pixelImgD >> 8) & 0xFF;
            Bd = pixelImgD & 0xFF;
            Anaglyphe.pixels[i] = color(Rg, Gd, Bd);
        }
    }
    photo_D.updatePixels();    photo_G.updatePixels();    Anaglyphe.updatePixels();

    image(Anaglyphe, 0, 0);
}
```

140

140

- Œil Gauche : Rouge = « Blanc » (Clair) et Vert/Bleu = « Noir » (foncé)
- Œil Droit : Rouge = « Noir » (foncé) et Vert/Bleu = inchangés

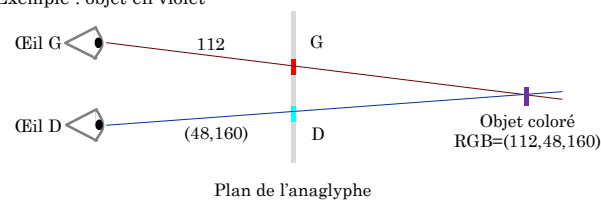


<https://www.david-romeuf.fr/3D/Anaglyphes/BonCoupleEL/BonCoupleEcranLunettesAnaglyphe.html>

141

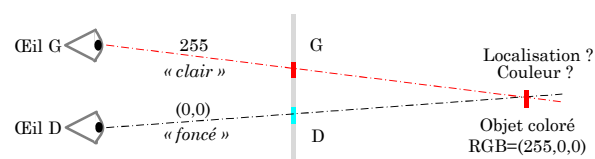
141

- Situation normale : objet coloré en couleurs non primaires
Exemple : objet en violet



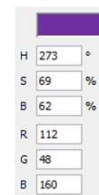
Plan de l'anaglyphe

- **Objet avec une couleur primaire (couleur « non stéréoscopique »)**
Exemple : objet en rouge pur



Plan de l'anaglyphe

Violet



142

142

IX.5 – Création anaglyphe améliorée

- Variantes dans la création d’images anaglyphes
 - Correction des couleurs
 - Modifier les pixels en couleurs primaires pour les rendre « stéréoscopiques »
 - Principe : prendre « un peu d’informations » (composantes RGB) de l’autre image
 - Exemple :
"A uniform metric for anaglyph computation", 2006, Z. Zhang, F. Mc Allister, Univ. Caroline du Nord - USA

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix}_{Image\ Anaglyphe} = \begin{pmatrix} \mathbf{0.4154} & 0.471 & 0.1669 & -0.0109 & -0.0364 & -0.006 \\ -0.0458 & -0.0484 & -0.0257 & 0.3756 & \mathbf{0.7333} & 0.0111 \\ -0.0547 & -0.0615 & 0.0128 & -0.0651 & -0.1287 & \mathbf{1.2971} \end{pmatrix} \times \begin{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}_{Image\ Gauche} \\ \begin{pmatrix} R \\ G \\ B \end{pmatrix}_{Image\ Droite} \end{pmatrix}$$

- Inconvénient :
 - augmente la diaphotie (à noter les soustractions de « Deghost » pour limiter le phénomène)
- Avantages :
 - Variantes de gris conservées
 - Couleurs obtenues correctes et non choquantes
 - Image globale légèrement plus sombre
 - Peau humaine un peu trop jaune/vert
 - Bleu du ciel trop bleu

143

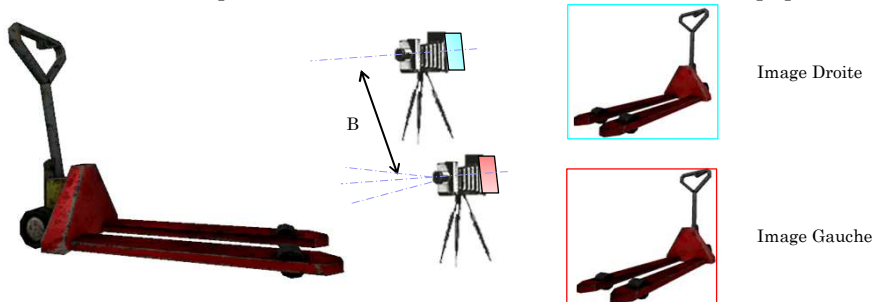
IX.6 – Scènes virtuelles stéréoscopiques

■ Objectif

- Visualiser le « virtuel » en stéréoscopie (en relief)
- Pas à partir de photos réelles... mais à partir des photos **virtuelles**

■ Principe

- Il faut tout d'abord dessiner (modéliser) une scène virtuelle
- Positionner deux appareils photos virtuels, écartés d'une « certaine » distance (base stéréoscopique B) paramétrable, ou non
- Orienter (ou non) les appareils pour provoquer une convergence
- Utiliser les photos virtuelles comme base de rendu stéréoscopique



145

145

IX.6 – Modélisation de scènes virtuelles dans Processing

■ Primitives élémentaires...

- de dessin 3D (primitives 3D de tracé) :
 - `box()`, `sphere()`, `loadShape()`, `shape()`, ... , `fill()`, `stroke()`, ...
- de changement de repère et transformations :
 - `translate()`, `rotate()`, `rotateX()`, `pushMatrix()`, `popMatrix()`, ...

■ ... à appliquer sur **deux** buffers PGraphics (un pour chaque œil)

- Un élément PGraphics permet le tracé « hors écran »
 - Le dessin est effectué en mémoire, mais la restitution sur l'écran n'est pas faite
- Primitives de gestion des PGraphics
 - Création : `PGraphics Graphique = createGraphics(300, 300, P3D);`
 - Encadrer les primitives classiques de `beginDraw()` et de `endDraw()`
 - Récupération de l'image équivalente : `Graphique.get();`
 - Cette action correspond à une « prise de photo »
 - avec l'appareil photo par défaut dans Processing

146

146

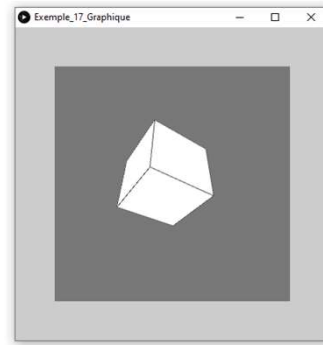
IX.7 – Exemple 17 : Dessin sur buffer PGraphics

```
PGraphics Graphique; // Buffer pour trace
PImage ImageEquivalente; // Tableau de pixels pour prise de photo

void setup() {
  size(400, 400, P3D);
  Graphique = createGraphics(300, 300, P3D);
  ImageEquivalente = createImage(300, 300, RGB);
}

void draw() {
  // Modelisation de la scene
  Graphique.beginDraw();
  Graphique.background(120);
  Graphique.stroke(80);
  Graphique.translate(140, 140);
  Graphique.rotateX(-mouseY/100.0);
  Graphique.rotateY(-mouseX/100.0);
  Graphique.box(80);
  Graphique.endDraw();

  // Affichage de la scene
  // image(Graphique, 50, 50); // Possible d'afficher directement
  // ou bien en recuperant le tableau de pixels (prise de photo)
  ImageEquivalente = Graphique.get();
  image(ImageEquivalente, 50, 50);
}
```

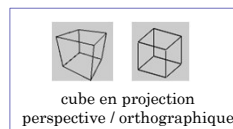
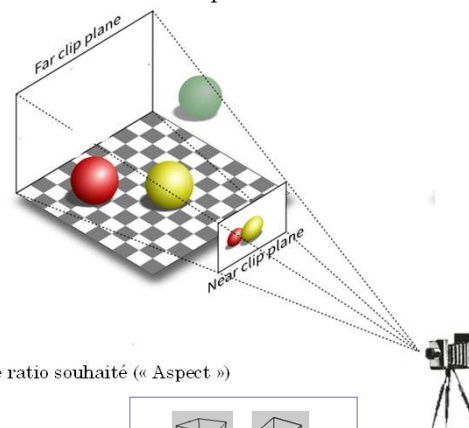


147

147

IX.8 – Configuration spécifique des caméras dans Processing

- Objectif
 - Configurer manuellement les deux caméras servant à la prise de vue des deux images Gauche et Droite
- Modèle d'une caméra
 - FOV et Frustum
 - Field Of View : angle d'ouverture vertical du point de vue
Généralement : $FOV = 30^\circ (= \pi / 6)$
 - Recul de la caméra (axe Z)
 $CameraZ = (HeightImg) / \tan(FOV / 2.0);$
 - Positions en Z des plans de « Clipping »
Généralement : facteur 1/10 et 10
 $NearClipZ = CameraZ / 10;$
 $FarClipZ = CameraZ * 10;$
 - Perspective / orthographique
 - Largeur de l'image WidthImg déduite par le ratio souhaité (« Aspect »)
 $Aspect = WidthImg / HeightImg;$
 - Configuration effective de la caméra
 - `perspective(FOV, Aspect, NearClipZ, FarClipZ);`



148

148

IX.8 – Configuration spécifique des caméras dans Processing

■ Modèle d'une caméra (suite)

□ Position et orientation spatiales

- Coordonnées de la caméra
- Coordonnées du centre de la scène
- (orientation du repère de la caméra)

□ Installation effective de la caméra

- `camera(eyeX, eyeY, eyeZ, centerX, centerY, centerZ, upX, upY, upZ)`
 - // eyeX float: x-coordinate for the eye
 - // eyeY float: y-coordinate for the eye
 - // eyeZ float: z-coordinate for the eye
 - // centerX float: x-coordinate for the center of the scene
 - // centerY float: y-coordinate for the center of the scene
 - // centerZ float: z-coordinate for the center of the scene
 - // upX float: usually 0.0, 1.0, or -1.0
 - // upY float: usually 0.0, 1.0, or -1.0
 - // upZ float: usually 0.0, 1.0, or -1.0

□ Exemple : deux caméras (une par PGraphics)

- `gDroit.camera(-demiIPD, 0, CameraZ, -centreSceneX, 0, 0, 0, 1.0, 0);`
`gGauche.camera(demiIPD, 0, CameraZ, centreSceneX, 0, 0, 0, 1.0, 0);`

149

149