# TP Limites de perception visuelle

Barbe Florian
Option Réalité Virtuelle – P1RV

19 Novembre 2025

## 1 Application de test 2D en C++ / OpenGL

### 1.1 Cahier des charges fonctionnel

**Question 1.1.** Quelles fonctionnalités minimales l'application doit-elle offrir ?

**Réponse.** L'application doit :

1. afficher soit l'image gauche seule, soit l'image droite seule, soit les deux en stéréo anaglyphe ;
2. être capable d'afficher, selon un paramètre de configuration, les objets suivants :
   — un point (modélisé par un disque plein) ;
   — une ligne verticale ;
   — une ligne horizontale ;
   — une image (texture) ;
3. proposer les interactions clavier suivantes :
   — augmenter/diminuer la *parallaxe horizontale* $d_x$ avec un pas réglable ;
   — augmenter/diminuer la *parallaxe verticale* $d_y$ avec un pas réglable ;
   — changer le type d'objet affiché (point, ligne verticale, ligne horizontale, image) ;
   — quitter l'application.

### 1.2 Conception de la scène stéréo

**Question 1.2.** Comment organiser la scène pour obtenir image gauche, image droite et image anaglyphe ?

**Réponse.** On adopte l'organisation suivante :

— On travaille dans une fenêtre OpenGL de taille $(W, H)$ en pixels, avec un repère 2D normalisé ou en coordonnées pixels.
— Pour l'**image gauche**, on dessine l'objet centré en $(x_0, y_0)$.
— Pour l'**image droite**, on dessine le même objet translaté de $(d_x, d_y)$ par rapport à l'image gauche, ce qui donne typiquement :

$$(x_{\text{gauche}}, y_{\text{gauche}}) = (x_0, y_0), \qquad (x_{\text{droite}}, y_{\text{droite}}) = (x_0 + d_x, y_0 + d_y).$$

— En mode **stéréo anaglyphe**, on dessine successivement la scène vue par l'œil gauche puis par l'œil droit, en filtrant les composantes couleur avec les masques de couleur adaptés (rouge/cyan ou rouge/bleu selon les lunettes).

### 1.3 Mappage des interactions clavier

**Question 1.3.** Proposer un mappage clavier simple pour les interactions demandées.

**Réponse.** Un exemple de mappage possible :

— ←/→ : diminuer/augmenter la parallaxe horizontale $d_x$ ;

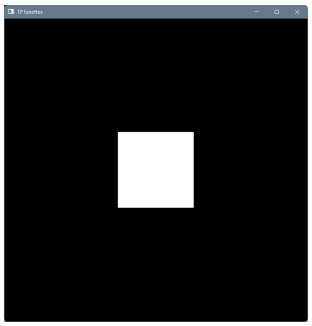

FIGURE 1 – Parallaxe horizontale nulle

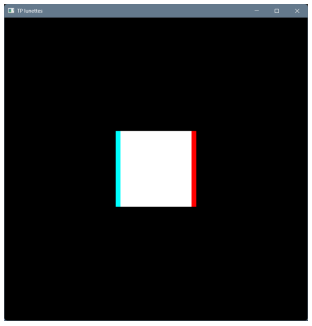

FIGURE 2 – Illustration de l'action de la touche ← : parallaxe horizontale négative

— ↓/↑ : diminuer/augmenter la parallaxe verticale $d_y$ ;

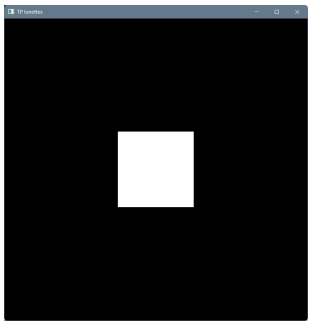

FIGURE 3 – Parallaxe verticale nulle

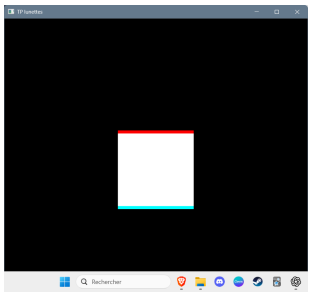

FIGURE 4 – Illustration de l'action de la touche ↑ : parallaxe verticale positive

— 0 : type d'objet point

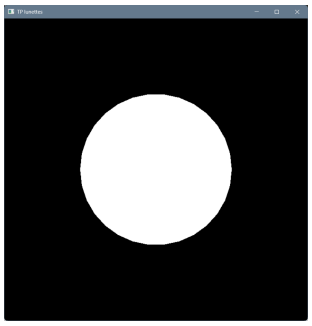

FIGURE 5 – Illustration du point

— `1` : type d'objet ligne verticale

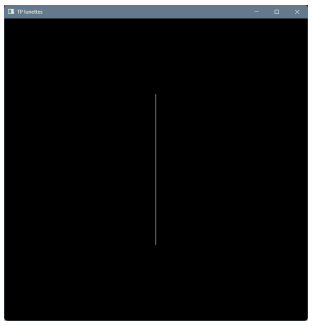

FIGURE 6 – Illustration de la ligne verticale

— `2` : type d'objet ligne horizontale

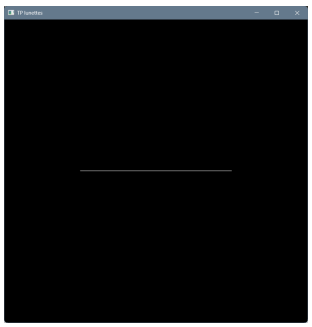

FIGURE 7 – Illustration de la ligne horizontale

— `3` : type d'objet carré

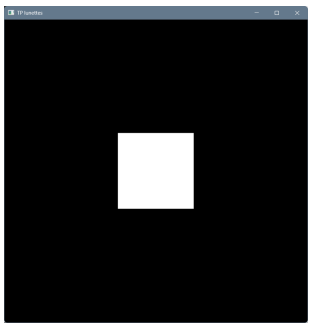

FIGURE 8 – Illustration du carré

— `4` : type d'objet théière

FIGURE 9 – Illustration de la théière

— `G` : forcer l'affichage de l'image gauche seule ;

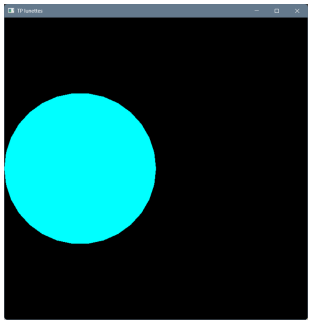

FIGURE 10 – Affichage de l'image gauche seule

— `D` : forcer l'affichage de l'image droite seule ;

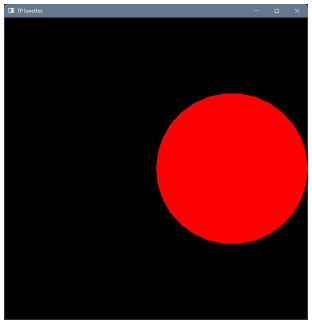

FIGURE 11 – Affichage de l'image droite seule

— `S` : afficher en stéréo anaglyphe ;

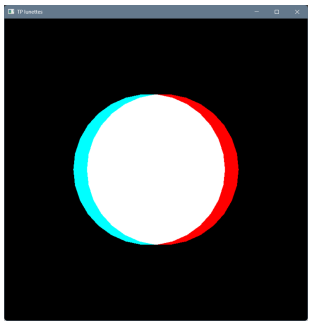

FIGURE 12 – Affichage stéréo anaglyphe

— `Esc` : quitter l'application.

# 2 Mesures et calculs préliminaires

## 2.1 Distance oeil–écran et champ de vision horizontal

**Question 2.1.1.** Comment mesurer la distance $d_s$ entre les yeux et l'écran ?

**Réponse.** On se place en position de travail normale, en posture confortable, puis on mesure la distance entre le plan des yeux et le plan de l'écran, à l'aide d'un mètre ruban. On note cette distance $d_s$ (en mètres). Par exemple, pour moi :

$$d_s^{(A)} \approx 0{,}65\,\text{m},$$

et pour un autre sujet :

$$d_s^{(B)} \approx 0{,}72\,\text{m}.$$

Ces valeurs sont à adapter aux mesures réelles.

**Question 2.1.2.** Comment calculer le champ de vision horizontal $FOV_h$ par rapport à l'écran ?

**Réponse.** On note $L$ la largeur physique de la zone utile de l'écran (en mètres). Le champ de vision horizontal $FOV_h$ vu depuis les yeux est l'angle sous lequel est vue cette largeur :

$$FOV_h = 2\arctan\left(\frac{L}{2d_s}\right),$$

où $FOV_h$ est exprimé en radians. Pour l'exprimer en degrés :

$$FOV_h(^\circ) = \frac{180}{\pi}\,2\arctan\left(\frac{L}{2d_s}\right).$$

Avec une largeur typique d'écran $L = 0{,}53\,\text{m}$ (écran 24 pouces) et $d_s^{(A)} = 0{,}65\,\text{m}$, on obtient :

$$FOV_h^{(A)} \approx \frac{180}{\pi}\,2\arctan\left(\frac{0.53}{2\times 0.65}\right) \approx 44{,}4^\circ.$$

## 2.2 Taille angulaire d'un pixel

**Question 2.1.3.** Comment calculer la taille $s$ d'un pixel à l'écran en minutes d'angle ?

**Réponse.** On note $N_x$ le nombre de pixels horizontaux de l'écran. La largeur d'un pixel vaut :

$$p = \frac{L}{N_x} \quad \text{(en mètres)}.$$

L'angle sous lequel est vu un pixel au centre de l'écran vaut, pour un petit angle :

$$s_{\text{rad}} \approx \frac{p}{d_s}.$$

En degrés :

$$s(^\circ) = s_{\text{rad}} \frac{180}{\pi} = \frac{p}{d_s} \frac{180}{\pi},$$

et en minutes d'angle :

$$s(\text{minutes}) = s(^\circ) \times 60 = \frac{p}{d_s} \frac{180}{\pi} \times 60.$$

Exemple : pour $L = 0{,}53\,\text{m}$, $N_x = 1920$, on a $p \approx 2.76 \times 10^{-4}\,\text{m}$ et, avec $d_s^{(A)} = 0{,}65\,\text{m}$,

$$s_{\text{rad}} \approx 4.25 \times 10^{-4}\,\text{rad}, \quad s(\text{minutes}) \approx 1{,}46'.$$

## 2.3 Tracé des courbes $FOV_h(d_s)$ et $s(d_s)$

**Question 2.1.4.** Comment obtenir les courbes $FOV_h(d_s)$ et $s(d_s)$ ?

**Réponse.** On fait varier $d_s$ sur un intervalle réaliste, par exemple $d_s \in [0.3, 1.0]\,\text{m}$, et on calcule :

$$FOV_h(d_s) = \frac{180}{\pi} 2 \arctan\left(\frac{L}{2d_s}\right),$$

$$s(d_s) = \frac{L}{N_x d_s} \frac{180}{\pi} \times 60.$$

On compile ensuite ces fonctions sur Python pour tracer les courbes correspondantes.

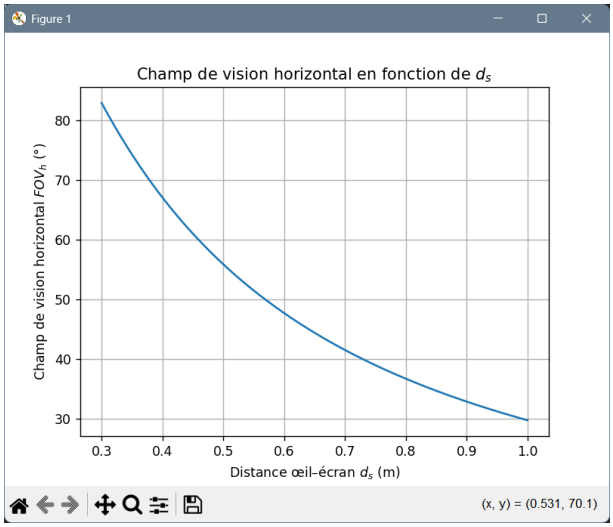


FIGURE 13 – Champ de vision horizontal $FOV_h$ en fonction de la distance oeil–écran $d_s$.

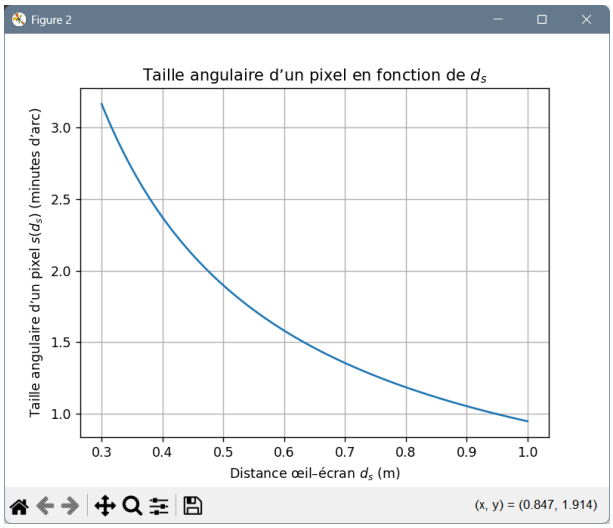


FIGURE 14 – Taille angulaire d'un pixel en fonction de la distance oeil–écran $d_s$.

Le champ de vision diminue quand $d_s$ augmente, tandis que la taille angulaire d'un pixel diminue également avec $d_s$ (les pixels « se rétrécissent » du point de vue de l'œil).

## 2.4 Zone de fusion en fonction de la distance de convergence

**Question 2.1.5.** Comment représenter la zone de fusion possible en fonction de la distance de convergence, en utilisant la contrainte $|\beta - \alpha| < 1.5°$ ?
**Réponse.** La disparité rétinienne d'un point situé à une distance $z$ des yeux est donnée, dans un modèle binoculaire simplifié, par :

$$|\beta - \alpha|(z, e) \;=\; 2 \arctan\!\left(\frac{e}{2z}\right),$$

où $e$ désigne l'écart interoculaire (typiquement $e \simeq 65\,\text{mm}$). La condition de fusion $|\beta - \alpha| < 1.5°$ permet de déterminer les configurations géométriques pour lesquelles le système visuel peut fusionner les deux images.

Pour représenter cette zone, nous avons évalué la fonction $|\beta - \alpha|(z, e)$ sur une grille bidimensionnelle couvrant :

$$z \in [0.2, 5.0]\ \text{m}, \qquad e \in [40, 80]\ \text{mm}.$$

La figure ci-dessous montre la surface tridimensionnelle $(z, e) \mapsto |\beta - \alpha|$, avec un code couleur distinguant explicitement :

— la **zone de non-fusion** (en rouge), où la disparité dépasse $1.5°$ ;
— la **zone de fusion** (en vert), correspondant à $|\beta - \alpha| < 1.5°$ ;
— le **plan horizontal** $|\beta - \alpha| = 1.5°$ (en violet), servant de frontière entre les deux régions.

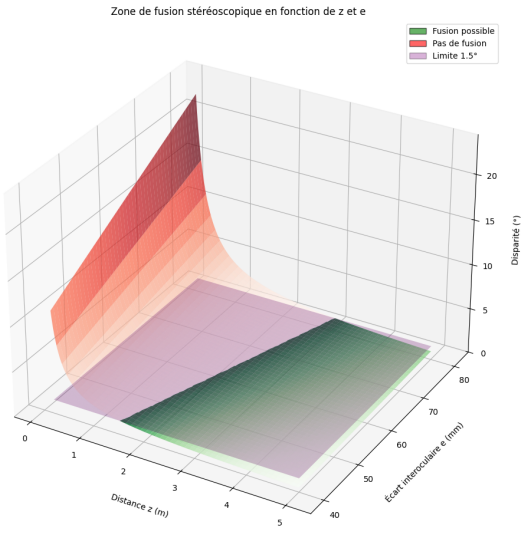


FIGURE 15 – Zone de fusion stéréoscopique en fonction de la distance $z$ et de l'écart interoculaire $e$.

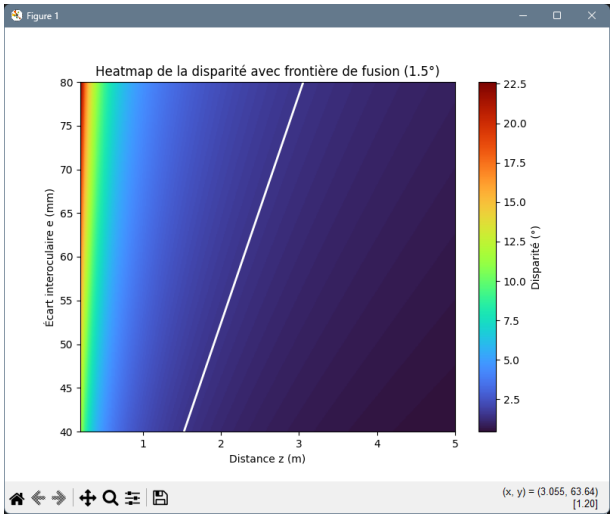


FIGURE 16 – Heatmap de la disparité $|\beta - \alpha|$ en fonction de la distance $z$ et de l'écart interoculaire $e$. La courbe blanche représente la limite de fusion à 1.5°.

Cette représentation met en évidence que, pour un écart interoculaire standard ($e \simeq 65$ mm), la disparité reste inférieure au seuil de 1.5° uniquement pour des distances relativement grandes ($z \gtrsim 2.5$ m). De plus, la zone de fusion se réduit lorsque $e$ augmente, illustrant le fait qu'un plus grand écart interoculaire accentue la disparité rétinienne.

En résumé, la zone de fusion correspond à l'ensemble des couples $(z, e)$ pour lesquels la disparité se situe sous la limite physiologique de 1.5° ; elle apparaît graphiquement comme la région verte située sous le plan seuil.

# 3 Parallaxe et disparité rétinienne

## 3.1 Lien entre parallaxe à l'écran et disparité angulaire

**Question 2.2.1.** Établir les équations qui relient la parallaxe à l'écran $(d_x, d_y)$ à la disparité rétinienne exprimée en angle.

**Réponse.** La parallaxe à l'écran correspond à un décalage en pixels (ou en mètres sur l'écran) entre l'image vue par l'œil gauche et l'œil droit.

On note :

— $d_x$ : parallaxe horizontale à l'écran (en mètres) ;

— $d_y$ : parallaxe verticale à l'écran (en mètres) ;

— $d_s$ : distance oeil–écran (en mètres).

Pour la composante horizontale, l'angle sous lequel est vue la séparation horizontale $d_x$ vaut, pour de petits angles :

$$\delta_x^{(\mathrm{rad})} \approx \frac{d_x}{d_s}.$$

En degrés :

$$\delta_x(^\circ) \approx \frac{d_x}{d_s} \cdot \frac{180}{\pi}.$$

De même pour la composante verticale :

$$\delta_y^{(\mathrm{rad})} \approx \frac{d_y}{d_s}, \qquad \delta_y(^\circ) \approx \frac{d_y}{d_s} \cdot \frac{180}{\pi}.$$

Si l'on travaille directement en pixels, avec un pas de pixel $p = L/N_x$, on a :

$$d_x = n_x\, p, \qquad \delta_x(^\circ) \approx \frac{n_x p}{d_s} \cdot \frac{180}{\pi},$$

où $n_x$ est le nombre de pixels de décalage entre les deux images.

### 3.2 Mesures expérimentales de disparité supportable

**Question 2.2.2.** Comment mesurer la disparité rétinienne maximale supportable en continu pour les différents objets ?

**Réponse.** La procédure est la suivante :

1. On se place dans la configuration d'observation standard (distance $d_s$ mesurée).
2. On initialise la parallaxe horizontale $d_x = 0$ (objet superposé pour les deux yeux).
3. On choisit un type d'objet (point, ligne verticale, ligne horizontale, image).
4. On augmente progressivement $d_x$ par pas réguliers (par exemple 1 pixel, puis 2, etc.) en laissant le temps au système visuel de fusionner les images.
5. On note la valeur $d_{x,\max}$ au-delà de laquelle la fusion devient impossible ou trop inconfortable (double vision, fatigue, etc.).
6. On convertit $d_{x,\max}$ en disparité angulaire via la relation précédente :

$$\delta_{x,\max}(^\circ) \approx \frac{d_{x,\max}}{d_s} \cdot \frac{180}{\pi}.$$

On répète cette expérience pour chaque type d'objet. On obtient un tableau de valeurs du type :

| Objet | $d_{x,\max}$ (pixels) | $\delta_{x,\max}$ (°) | Commentaires |
|---|---|---|---|
| Point | 12 | 0.29 | Fusion facile, repère ponctuel très précis. |
| Ligne verticale | 14 | 0.34 | Référence claire pour la disparité horizontale, fusion encore confortable. |
| Ligne horizontale | 8 | 0.19 | Moins discriminant en horizontal, la double vision apparaît plus tôt. |
| Image | 16 | 0.39 | Beaucoup de détails, aide à la fusion malgré une disparité plus forte. |

**Question 2.2.3.** Comment mesurer la disparité rétinienne maximale supportable après fermeture des yeux ?

**Réponse.** On suit la même démarche, mais en modifiant la condition expérimentale :

1. On règle la parallaxe $d_x$ à une valeur donnée (non connue de l'observateur).
2. On demande au sujet de fermer les yeux quelques secondes (afin de « réinitialiser » la fusion).
3. Au moment où le sujet rouvre les yeux, on observe s'il parvient à fusionner la scène ou non.
4. On ajuste $d_x$ pour trouver la valeur maximale à partir de laquelle la fusion n'est plus possible lors de la réouverture.

On obtient ainsi une disparité maximale supportable en *accès direct*, qui est souvent plus faible que la disparité atteignable par adaptation progressive.

Dans notre configuration ($d_s \approx 0{,}65\,\text{m}$), nous avons obtenu après fermeture des yeux :
— point : $d_{x,\max} \approx 10$ pixels, soit $\delta_{x,\max} \approx 0{,}24°$ ;
— ligne verticale : $d_{x,\max} \approx 12$ pixels, soit $\delta_{x,\max} \approx 0{,}29°$ ;
— ligne horizontale : $d_{x,\max} \approx 6$ pixels, soit $\delta_{x,\max} \approx 0{,}15°$ ;
— image : $d_{x,\max} \approx 14$ pixels, soit $\delta_{x,\max} \approx 0{,}34°$.

On observe que les valeurs maximales sont un peu plus faibles qu'en adaptation progressive, ce qui est cohérent avec le fait que le système visuel dispose de moins de temps pour s'habituer à la disparité.

**Question 2.2.4.** Comment procéder pour la parallaxe verticale ?

**Réponse.** La méthodologie est identique, en jouant cette fois sur $d_y$ plutôt que sur $d_x$ :
— on augmente progressivement la parallaxe verticale $d_y$ ;
— on mesure la valeur maximale $d_{y,\max}$ pour laquelle la fusion reste possible (en continu ou après fermeture des yeux).

En pratique, la tolérance à la disparité verticale est nettement plus faible que pour la disparité horizontale : de petites erreurs d'alignement vertical produisent rapidement un inconfort important.

Dans notre expérience, la disparité verticale maximale supportable est restée très faible :
— en adaptation progressive, on atteint typiquement $d_{y,\max} \approx 4$ pixels, soit $\delta_{y,\max} \approx 0{,}10°$ ;
— après fermeture des yeux, la fusion n'est plus possible au-delà de 2 à 3 pixels, soit $\delta_{y,\max} \approx 0{,}05$–$0{,}07°$.

Ces valeurs sont nettement inférieures aux disparités horizontales, ce qui confirme que le système visuel tolère très mal les erreurs d'alignement vertical.

# 4 Bonus : acuité stéréoscopique (non obligatoire)

## 4.1 Principe expérimental

**Question 3.1.** Quel est le principe de la mesure d'acuité stéréoscopique ?

**Réponse.** L'acuité stéréoscopique correspond à la plus petite différence de profondeur $\Delta z$ entre deux objets qu'un sujet est capable de percevoir en stéréoscopie. Pour la mesurer :

1. On configure une scène 3D dans OpenGL avec deux caméras modélisant les yeux, séparées par la distance interoculaire $e \approx 65\,\text{mm}$.
2. On choisit un plan de convergence coïncidant avec le plan de l'écran : un premier objet est placé sur ce plan.
3. Un second objet est positionné à une profondeur $z = z_c + \Delta z$, avec $\Delta z$ contrôlé.

4. On fait varier $\Delta z$ (en avant ou en arrière du plan de l'écran) et on demande au sujet s'il perçoit une différence de profondeur.

On ajuste $\Delta z$ par dichotomie (ou par pas décroissants) pour trouver la plus petite valeur encore détectable, ce qui donne une estimation de l'acuité stéréoscopique du sujet, exprimée en mm ou en disparité angulaire.

### 4.2 Configuration des caméras

**Question 3.2.** Comment configurer les caméras OpenGL pour simuler une observation binoculaire réaliste ?

**Réponse.** On adopte une configuration standard :

— Deux caméras dont les positions sont $\left(-\frac{e}{2}, 0, 0\right)$ et $\left(\frac{e}{2}, 0, 0\right)$ dans un repère centré au milieu des yeux.

— Les deux caméras regardent vers un même point de convergence situé sur le plan de l'écran (par exemple $(0, 0, z_c)$).

— Les matrices de projection sont ajustées de sorte que la projection sur le plan écran soit cohérente avec la géométrie réelle (distance de l'écran, taille physique, etc.).

Les rendus gauche et droit sont ensuite combinés en anaglyphe selon la méthode décrite à la section suivante.

## 5 Affichage en stéréo anaglyphe

### 5.1 Principe du masquage couleur

**Question 4.1.** Quel est le principe général de l'affichage stéréo anaglyphe avec OpenGL ?

**Réponse.** Le principe consiste à coder l'image destinée à l'œil gauche dans certaines composantes couleur (par exemple le rouge) et l'image destinée à l'œil droit dans d'autres composantes (par exemple le bleu et le vert, formant le cyan). Les lunettes anaglyphes filtrent ensuite les composantes couleur de sorte que chaque œil ne voie que l'image qui lui est dédiée.

En pratique, on procède en deux passes de rendu :

1. on dessine la scène vue par l'œil gauche en n'autorisant que certaines composantes couleur (masque couleur 1) ;
2. on dessine la scène vue par l'œil droit en n'autorisant que les composantes complémentaires (masque couleur 2), en ajoutant les couleurs de manière additive.

### 5.2 Pseudo-code OpenGL

**Question 4.2.** Donner le pseudo-code OpenGL proposé pour l'affichage anaglyphe.

**Réponse.** Le rendu anaglyphe utilise deux passes : une pour l'œil gauche, une pour l'œil droit, en filtrant les composantes couleur.

```
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

# ---- Première passe : œil gauche ----
glColorMask(1, 0, 0, 1);          # garder uniquement le rouge
# config caméra œil gauche
# draw_scene_left();

glClear(GL_DEPTH_BUFFER_BIT);
glEnable(GL_BLEND);
glBlendFunc(GL_ONE, GL_ONE);
```

```
# ---- Deuxième passe : œil droit ----
glColorMask(0, 1, 1, 1);          # garder vert + bleu = cyan
# config caméra œil droit
# draw_scene_right();
```