ECOLE CENTRALE DE NANTES

Option Réalité virtuelle
TP Limites de perception visuelle

27 septembre 2021

L'objectif de ce TP est de faire converger des éléments de trois cours différents (C++, image de synthèse et fondements de la réalité virtuelle) afin de se livrer à quelques expériences de perception visuelle.

1 APPLICATION DE TEST

L'objectif de cette partie est de construire une petite application graphique **2D** interactive en C++ (OpenGL) qui comporte les fonctionnalités suivantes :

1. Elle est capable d'afficher l'image gauche seule, l'image droite seule ou deux images en stéréo anaglyphe.
2. Cette fonction d'affichage devra être capable d'afficher plusieurs types d'objet selon le paramétrage de l'application : un point (représenté sous la forme d'un disque plein), une ligne verticale, une ligne horizontale ou un cube.
3. Les interactions possibles seront les suivantes :
   — augmenter et diminuer la parallaxe horizontale (le pas devra être réglable)
   — augmenter et diminuer la parallaxe verticale (le pas devra être réglable)
   — changer l'objet affiché
   — quitter l'application

Vous pouvez tester l'application en vérifiant que tout se passe bien en affichant seulement l'image gauche (respectivement droite) et les deux images simultanément avec et sans lunettes. Dans ce dernier cas, vous pouvez utiliser des couleurs différentes pour bien vérifier que la configuration de l'écran est OK.

La machine virtuelle Linux ne permet pas aujourd'hui l'affichage en stéréo, le TP doit donc être fait avec **Visual Studio sous Windows**.

2 Expériences de perception visuelle

2.1 Mesures et calculs préliminaires

Dans cette partie, nous allons chercher vos limites de perception visuelle. Commencez par mesurer ou calculer les paramètres suivants :

1. La distance d_s entre vos yeux et l'écran
2. Calculer votre champ visuel horizontal FOV_h par rapport à l'écran
3. Calculer la taille s d'un pixel à l'écran en minutes d'angle

Tracez les courbes $FOV_h(d_s)$ et $s(d_s)$ en faisant bien apparaître d_s.
On a vu en cours que la fusion était grosso modo possible lorsque la disparité rétinienne $|\beta - \alpha| < 1.5°$. Tracez sur une figure supplémentaire la zone de fusion possible en fonction de la distance de convergence.

2.2 Parallaxe

Établissez les équations qui relient la parallaxe à l'écran (d_x et d_y) à la disparité rétinienne exprimée sous forme angulaire.
Ensuite, mesurez les quantités suivantes pour chacun des objets prévus dans la partie précédente :

— la disparité rétinienne maximale que vous pouvez supporter en continu, c'est-à-dire en partant de $d_x = 0$ et en augmentant progressivement sa valeur
— la disparité rétinienne maximale que vous pouvez supporter en fermant les yeux quelques secondes et les rouvrant sur l'écran
— les mêmes mesures pour la parallaxe verticale

3 Bonus : acuité stéréoscopique

La mesure de l'acuité stéréoscopique consiste à mesurer la plus petite différence de profondeur perceptible par l'œil humain. Pour cela, vous devez configurer le programme précédent pour afficher des objets en 3D (configuration OpenGL) et configurer les caméras pour afficher en fonction de la distance interoculaire (vous pouvez prendre 65mm en première approximation) et un angle de convergence entre les yeux qui permet de fixer le point de convergence au niveau de l'écran.
Ensuite, vous pourrez afficher deux objets 3D, l'un dans le plan de l'écran et l'autre dont la distance au plan de l'écran sera progressivement réduite. Vous pourrez modifier les pas de votre algorithme pour déterminer la plus petite différence de profondeur entre les deux objets qui permet encore une distinction de profondeur.

ANNEXE : AFFICHAGE EN STÉRÉO ANAGLYPHE

Il existe plusieurs façons d'afficher des images en stéréo anaglyphe (vous pouvez chercher sur internet). Une des méthodes les plus simples est proposée par Paul Bourke [1] et utilise simplement les masques OpenGL. Le pseudo-code suivant suffit pour dessiner en stéréo anaglyphe (si les couleurs utilisées ne se recouvrent pas).

```
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

glColorMask(GL_TRUE, GL_FALSE, GL_FALSE, GL_TRUE);

// set camera for blue eye, red will be filtered.

// draw scene

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

glColorMask(GL_FALSE, GL_FALSE, GL_TRUE, GL_TRUE);

// set camera for red eye, blue will be filtered.

// draw scene
```

Les combinaisons exactes de masque sont à prendre dans le fichier fourni par Paul Bourke en fonction des lunettes disponibles. Si les couleurs se recouvrent, il faudra utiliser en plus les accumulation buffer comme mentionné dans l'exemple de Paul Bourke.

A RENDRE

Un compte-rendu complet de TP est attendu, pas seulement le code! Il doit comprendre les mesures demandées, les valeurs trouvées pour les deux membres du binôme le cas échéant ainsi que des commentaires et analyses sur les valeurs trouvées.
Le TP est à rendre avant le 5 novembre 2021 au soir, par mail à martin.guy@ec-nantes.fr. Le TP sera noté, et tout éventuel retard sera sanctionné.

1. http ://paulbourke.net/stereographics/anaglyph/