

INF2705 Infographie

Spécification des requis du système

Travail pratique 2

Stencils, caméras et nuanceurs

Table des matières

1	Introduction	2
1.1	But	2
1.2	Portée	2
1.3	Remise	2
2	Description globale	3
2.1	But	3
2.2	Contexte : Modélisation d'observation interstellaire	3
2.3	Travail demandé	5
3	Exigences	11
3.1	Exigences fonctionnelles	11
3.2	Exigences non fonctionnelles	11
A	Liste des commandes	12
B	Figures supplémentaires	13
C	Apprentissage supplémentaire	15

1 Introduction

Ce document décrit les exigences du TP2 « *Stencils, caméras et nuanceurs* » (Hiver 2023) du cours INF2705 Infographie.

1.1 But

Le but des travaux pratiques est de permettre à l'étudiant de directement appliquer les notions vues en classe.

1.2 Portée

Chaque travail pratique permet à l'étudiant d'aborder un sujet spécifique.

1.3 Remise

Faites la commande « `make remise` » ou exécutez/cliquez sur « `remise.bat` » afin de créer l'archive « **INF2705_remise_TPn.zip** » (ou .7z, .rar, .tar) que vous déposerez ensuite dans Moodle. (Moodle ajoute automatiquement vos matricules ou le numéro de votre groupe au nom du fichier remis.)

Ce fichier zip contient tout le code source du TP (`makefile`, `*.h`, `*.cpp`, `*.glsl`, `*.txt`).

2 Description globale

2.1 But

Le but de ce TP est de permettre à l'étudiant d'utiliser les plans de coupe et de se familiariser avec les fonctions de manipulation du tampon stencil telles que `glStencilFunc()` et `glStencilOp()`. Il permettra également de mettre en pratique la caméra synthétique, le nuanceur de géométrie et l'illumination d'une scène.

2.2 Contexte : Modélisation d'observation interstellaire

Vous avez été parmi les heureux élus à avoir contribuer au télescope spatial James Webb lors de votre dernier stage. Sous la supervision de l'ingénierie système principale du détecteur de guidage de précision (FGS), vous avez pu produire différentes modélisation 3D du télescope et grâce à votre professionnalisme et vos talents d'infographe, l'Agence Spatiale Canadienne vous contacte pour un projet spécial.

Les premières transmissions reçues du télescope spatial James Webb (JWST) ont mis en évidence l'existence d'un « trou de ver » proche d'Eris à l'intérieur de la Ceinture de Kuiper(figure 12).

Votre mandat se résume à produire un rendu 3D du trou de ver et les exoplanètes, ainsi qu'aider les chercheurs de l'ASC avec les simulations de sonde exploratrice en modélisant ce système solaire interstellaire.

Dans ce cours d'infographie, on remplacera les exoplanètes (qui sont difficiles à observer en réalité) par de superbe isocaèdre (ou un d20 pour les initiés). De plus, un trou de ver étant en théorie sphérique, nous allons le modéliser par un cube, ce qui nous simplifiera plusieurs étapes en réduisant le nombre de faces à dessiner. Les planètes auront des couleurs variées afin de les identifier plus aisément.

Le tout nous permettra de visualiser ce système intergalactique.

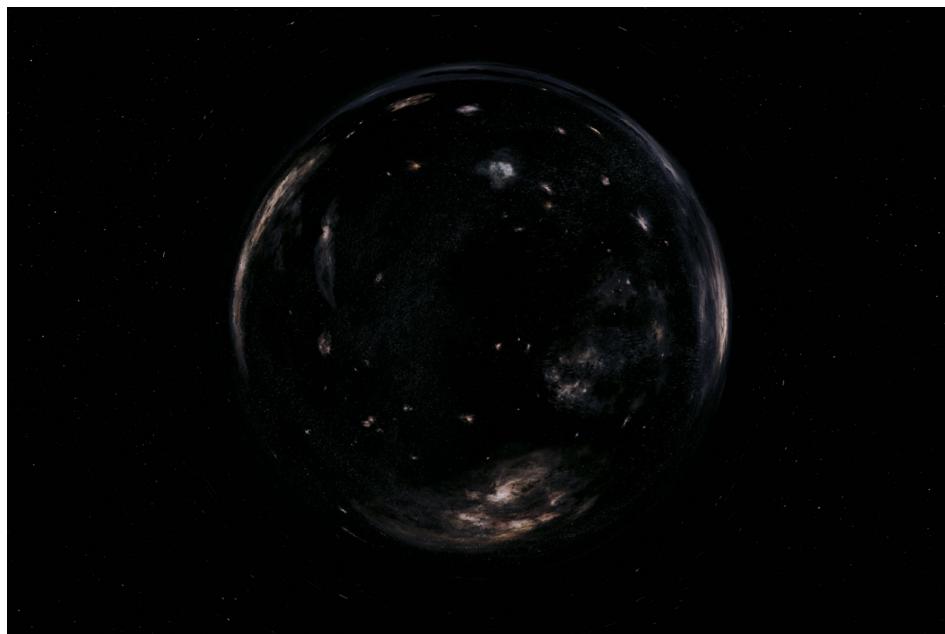


FIGURE 1 – Exemple de trou de ver (Interstellar)

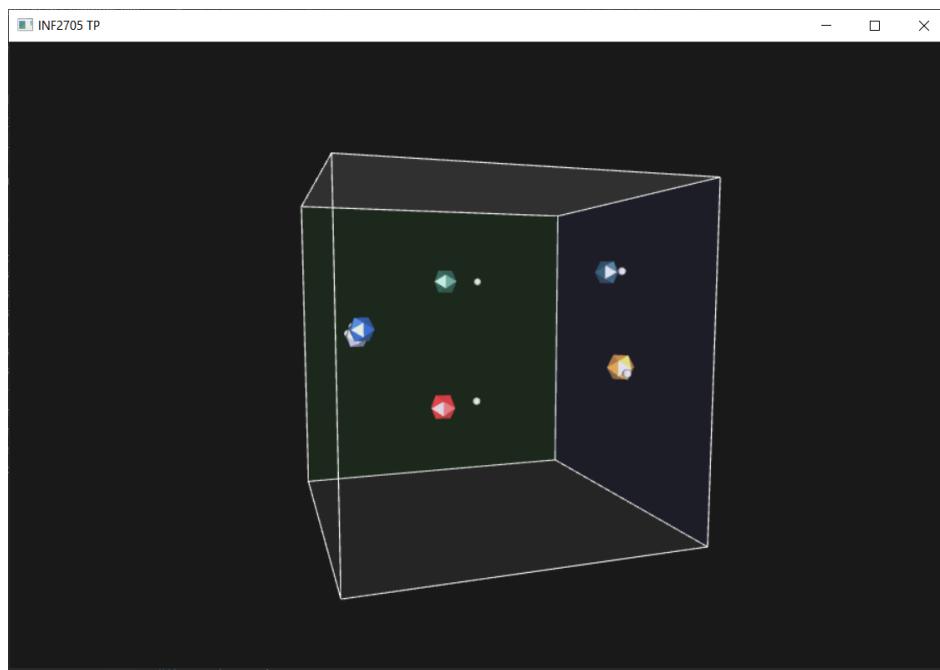


FIGURE 2 – Notre rendu final avec illumination

2.3 Travail demandé

Partie 1 : tampon de stencil

Les "parois" de notre trou de ver seront des "murs" entre les dimensions qui nous permettrons de voir les exoplanètes et les satellites en orbite. Afin d'avoir le rendu souhaité, nous devrons tracer la scène "derrière" chaque paroi (figure 3). Ensuite, nous devrons utiliser notre (nos) stencil pour ne pas afficher tout ce qui se trouve à l'extérieur de notre trou de ver.(figure 4)

Remarquez que vous aurez probablement à faire plusieurs appels aux différentes méthodes "afficher...()" pour obtenir ce résultat. Les stencils que nous utiliserons auront une valeur de *ref* différente les uns des autres pour s'assurer que l'on affiche pas notre scène en double, triple etc.

Autrement dit, vous pourrez utiliser 1,2,4,8,16,32 comme valeurs pour chaque paroi. Lorsque le stencil aura les valeurs appropriées, vous pourrez afficher les planètes en spécifiant le test du stencil par rapport aux parois.

Pour alléger le code, il n'y aura pas de planète qui sera "au-dessus/en dessous" du trou de ver (figure 11). Libre à vous de modifier le code pour en ajouter lorsque vous aurez terminer le TP !

Attention, n'utilisez pas `glClear()` entre l'affichage des parois, cette méthode doit être appellé seulement une fois au début de notre boucle d'affichage et pas à l'intérieur !

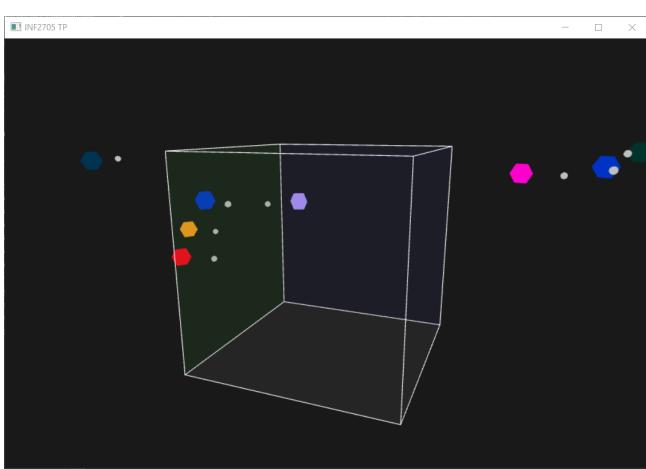


FIGURE 3 – Trou de ver sans stencil

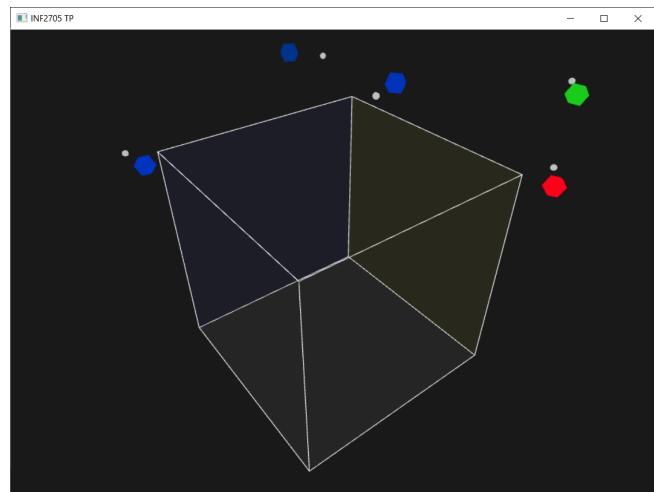


FIGURE 4 – Vue de haut sans stencil

Pour s'inspirer, vous pouvez consulter ces exemples sur gitlab :

<https://gitlab.com/ozell/inf2705-exemples/-/tree/master/04-StencilMiroir/> et
<https://gitlab.com/ozell/inf2705-exemples/-/tree/master/04-StencilTheiere/>.

Partie 2 : Point de vue d'une sonde spatiale

Positionnement de la caméra. Pour se pratiquer à bien positionner la caméra, on affichera le point de vue selon un satellite en rotation autour d'une planète, c'est-à-dire en positionnant la caméra légèrement en arrière de ce satellite. Ainsi, selon le choix d'une des touches 1 à 9, on affichera le point de vue d'un des neuf premières planètes de la liste, tandis que la touche 0 affichera le point de vue standard dans notre système solaire (figure 5). Noter que les cinq premières planètes sont facile à identifier par leur couleur (rouge, orangé , magenta , vert , violet pale), alors que les suivantes sont des variations de couleurs bleutées.

La stratégie pour positionner la caméra à une planète est de déplacer le repère de la façon habituelle jusqu'à la position voulue, de récupérer la matrice de modélisation courante, de l'inverser et de s'en servir comme matrice de visualisation. (C'est un simple changement de repère tel que vu en classe.)

La méthode `Theiere::obtenirMatriceCourante()` retournera la matrice de modélisation courante (`mtc`) représentant la position de la planète choisie, c'est-à-dire les déplacements pour amener le repère à la position voulue¹. Ensuite, la méthode `TrouDeVer::positionnerCamera()` calculera l'inverse de cette matrice de modélisation courante et s'en servira pour initialiser la matrice de visualisation : `matrVisu.setMatr(glm::inverse(glm::mat4(mtc)))`. Notez que le déterminant de la matrice de visualisation doit toujours être unitaire.²

Une planète se déplace en révolution autour de son astre ainsi qu'en rotation autour de son centre. Toutefois, puisque la NASA et l'ASC ne possèdent aucune information sur la composition des planètes ils voudraient que l'on modélise la vue à partir d'un satellite de la planète ou en étant dans un orbite géosynchrone (figure 6) Pour accomplir le tout, nous devrons faire quelques transformations supplémentaires pour obtenir la bonne `mtc` en amenant notre repère derrière le satellite et soit synchroniser son orbite avec le satellite ou la planète.

Pour bien faire les choses, il est préférable de récupérer la matrice courante de la planète et de corriger l'orientation du repère avant de l'inverser.

Les satellites sont tous à 3.5 unités du centre de la planète. De plus, selon les estimations des chercheurs, chaque planète possède une masse très différente que vous avez considérée lors de la mise à l'échelle de chaque planète à l'aide du facteur `taille`. Ces informations vous seront utiles pour appliquer les bonnes transformations au repère.

3

Nuanceur de géométrie. Il sera intéressant de débuter le nuanceur de géométrie en prenant les sommets reçus du nuanceur de sommets et de simplement les relayer au nuanceur de fragments. La semaine prochaine, nous utiliserons ces deux derniers nuanceurs pour ajouter une illumination aux planètes.

1. Pourquoi ne peut-on simplement pas sauvegarder et utiliser une copie de la matrice de modélisation produite lors de l'affichage de cette théière ?

2. Pour quelles raisons le déterminant de la matrice de visualisation ne serait pas égal à 1 ? Quel serait l'effet visuel ?

3. Pourquoi ne pas simplement faire une mise à l'échelle de `taille` et une translation fixe de 3.5 unités ?

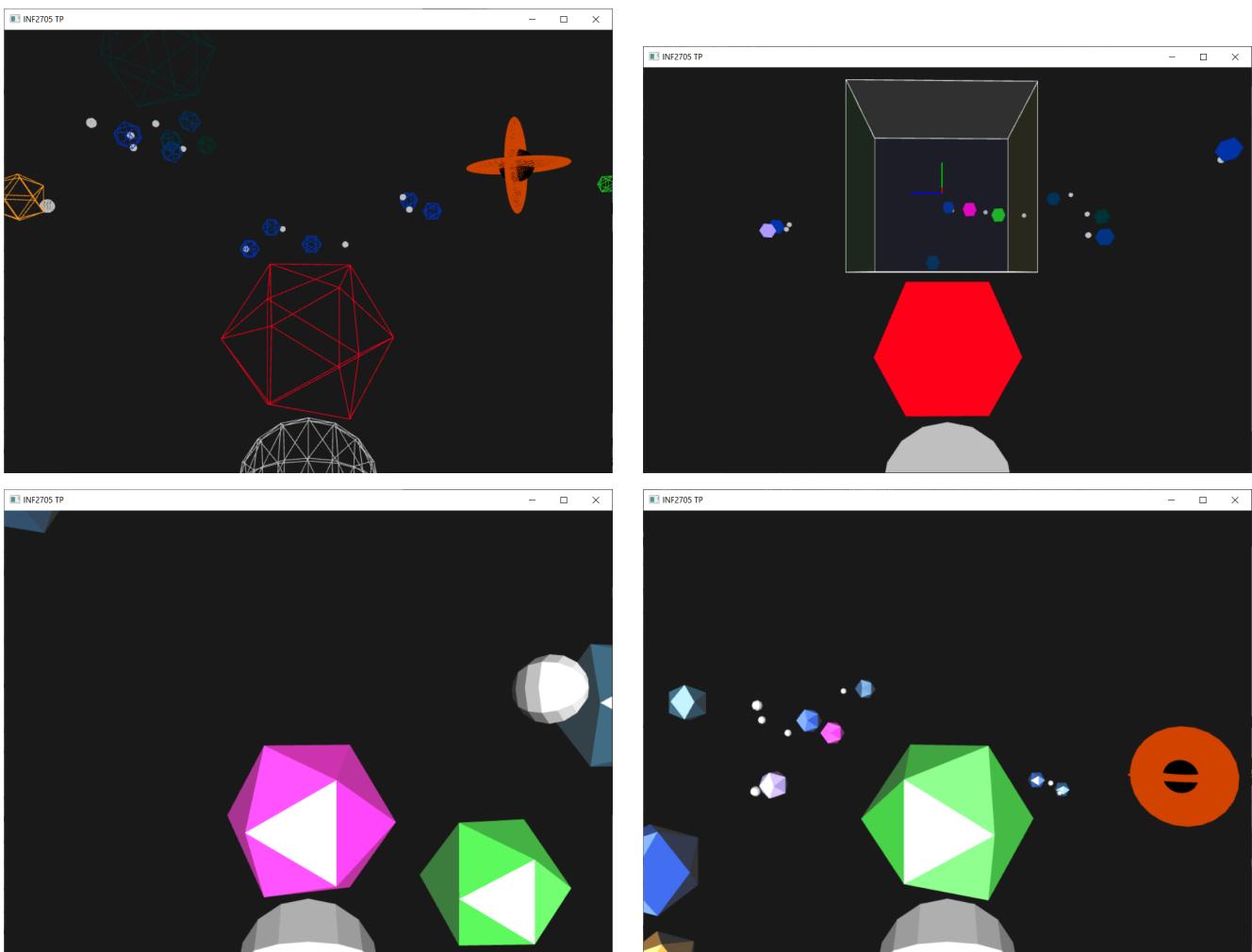


FIGURE 5 – Points de vue selon diverses planètes

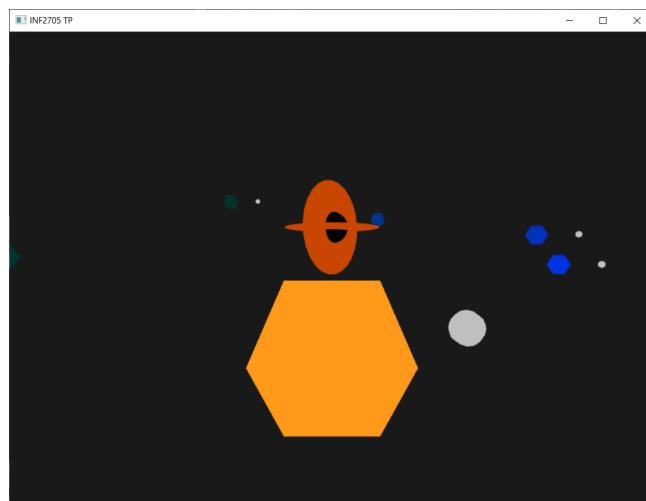


FIGURE 6 – En étant géosynchrone, notre caméra fixera le même point sur la planète et tourneras comme elle

Partie 3 : illumination et nuanceur de géométrie

Vous recevez plusieurs courriels cette semaine de la NASA. Il semblerait que le trou de ver soit en fait un trou noir ! Toutefois, ce trou noir est assez jeune et son disque d'accrétion (figure 9) est suffisamment large qu'il émet énormément de lumière. Afin d'améliorer le modèle vous décidez donc d'ajouter une illumination de Phong dans la scène à l'aide du nuanceur de fragments et de géométrie. Ce dernier servira à calculer la normale de chaque face en appliquant le produit vectoriel de deux arêtes de chaque triangle.

Ainsi, dans le nuanceur de géométrie que vous ajouterez, vous recevez les trois sommets de chaque triangle dans `gl_in[i].gl_Position`, $i=0,1,2$. Il sera donc facile de calculer le produit vectoriel entre deux arêtes du triangle pour obtenir un vecteur perpendiculaire à ces deux vecteurs (et à la surface du triangle), c'est-à-dire la normale (N) recherchée :

```
vec3 arete1 = ( Sommet1 - Sommet0 );
vec3 arete2 = ( Sommet2 - Sommet0 );
normale = cross( arete1, arete2 );
```

De plus, la source de lumière sera positionnée au loin dans la direction de la caméra afin de simplifier nos calculs et d'utiliser un vecteur constant :

```
lumiDir = vec3( 0, 0, 1 ).
```

Enfin, on considérera aussi que l'observateur est toujours placé au plan avant et nous utiliserons :

```
obsVec = vec3( 0, 0, 1 ).
```

Ces modifications pour la position de la lumière et de l'observateur sont souvent utilisées pour que les calculs d'illumination soient plus simples. Dans ce TP, nous utiliserons le nuanceur de géométrie pour imposer ces valeurs à chaque face, mais nous utiliserons quand même l'illumination complète de Phong dans le nuanceur de fragments, sans y chercher à simplifier les calculs.

Pour s'inspirer, vous pouvez utiliser les exemples d'illumination vus au cours :

<https://gitlab.com/ozell/inf2705-exemples/-/tree/master/06-IlluminationMiroir/> et
<https://gitlab.com/ozell/inf2705-exemples/-/tree/master/06-IlluminationTheieres/..>

Vous pouvez aussi utiliser l'exemple du nuanceur de géométrie vu au cours :

<https://gitlab.com/ozell/inf2705-exemples/-/tree/master/05-NuanceurGeometrie/>.

Note : Ce TP utilise des « *Uniform Buffer Object* » (UBO) afin de transférer en bloc les variables uniformes aux nuanceurs. L'usage des UBO est très semblable aux VBO et permet surtout que le passage des valeurs des variables uniformes aux nuanceurs soit beaucoup plus efficace. Dans ce TP, on utilise ainsi trois UBO qui correspondent aux trois blocs de variables uniformes utilisés dans les nuanceurs : `LightSource`, `FrontMaterial` et `LightModel`. Les trois blocs contiennent les variables uniformes servant à l'illumination. (Les noms de variable sont inspirés de OpenGL 2.x.)

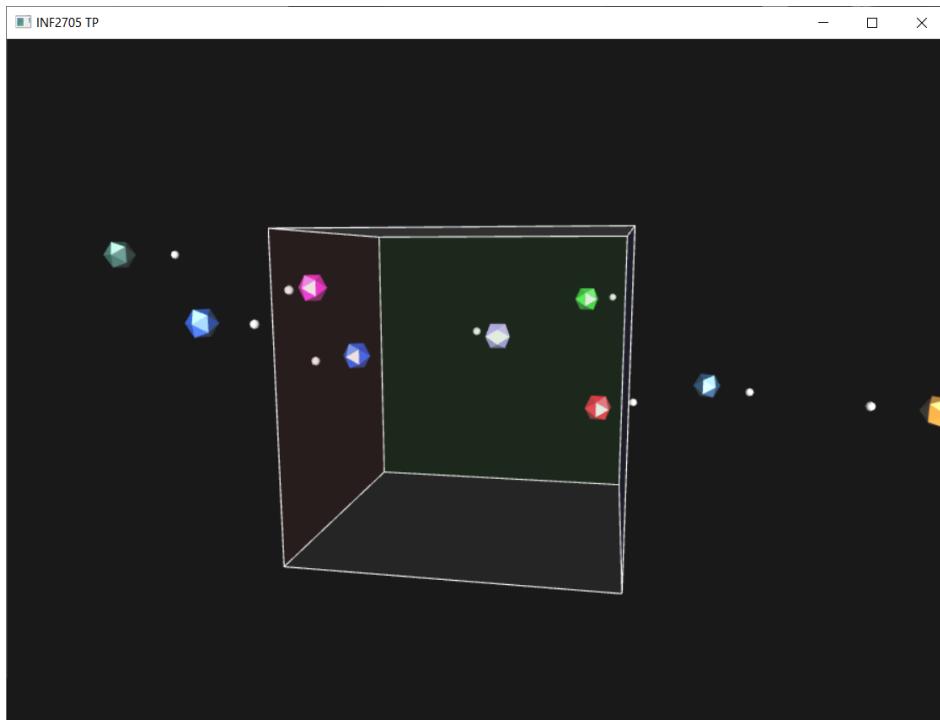
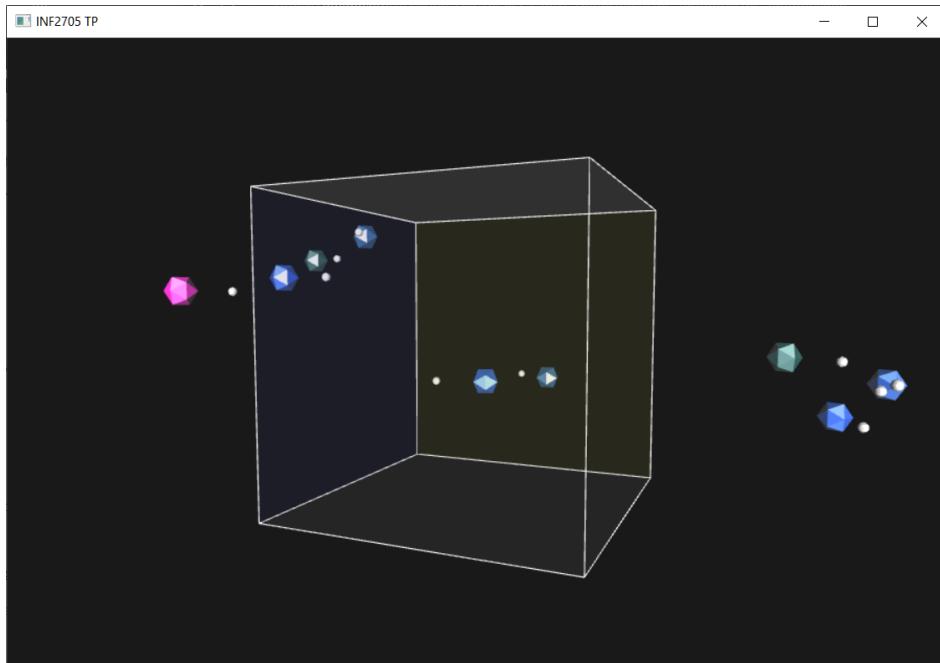


FIGURE 7 – Illumination avec des vecteurs normaux constants par face.

Partie 4 : daltonisme Monochromatique

Nous désirons favoriser l'inclusion de tous en infographie et puisque ce cours à un aspect manifestement visuel, nous nous intéressons donc au problème de daltonisme Monochromatique.

Certaines personnes peuvent en être atteintes, ce qui ne leur permet pas de voir clairement les couleurs. Ils n'ont qu'un seul pigment de cône fonctionnel. Ces cônes permettent de voir les couleurs alors que les bâtonnets leur permettent de voir les nuances de couleurs. Dans ce cas-ci, on parle de nuances de gris.⁴ Vous aurez à appliquer la formule suivante dans un des nuanceurs :

$$Y = 0,229R + 0,587G + 0,114B$$

Dans le modèle YUV, la partie Y représente la luminance ou l'intensité de lumière. La partie UV est reliée à la chrominance ou l'information sur la couleur. L'avantage de ce modèle est qu'il permet de faire des traitements sur l'intensité sans modifier la chrominance et vice-versa. Étant donné que les personnes atteintes de daltonisme Monochromatique ne peuvent voir les couleurs, nous nous intéresserons qu'à la partie Y dont la formule qui vous a été donnée. Vous devez l'appliquer sur chaque canal de couleur de la même manière, donc à "r", "g" et "b" dans rgba, mais pas au canal "a" qui représente l'opacité. La figure 8 présente l'effet monochromatique sur les couleurs normales sans illumination (en haut) et avec illumination (en bas).

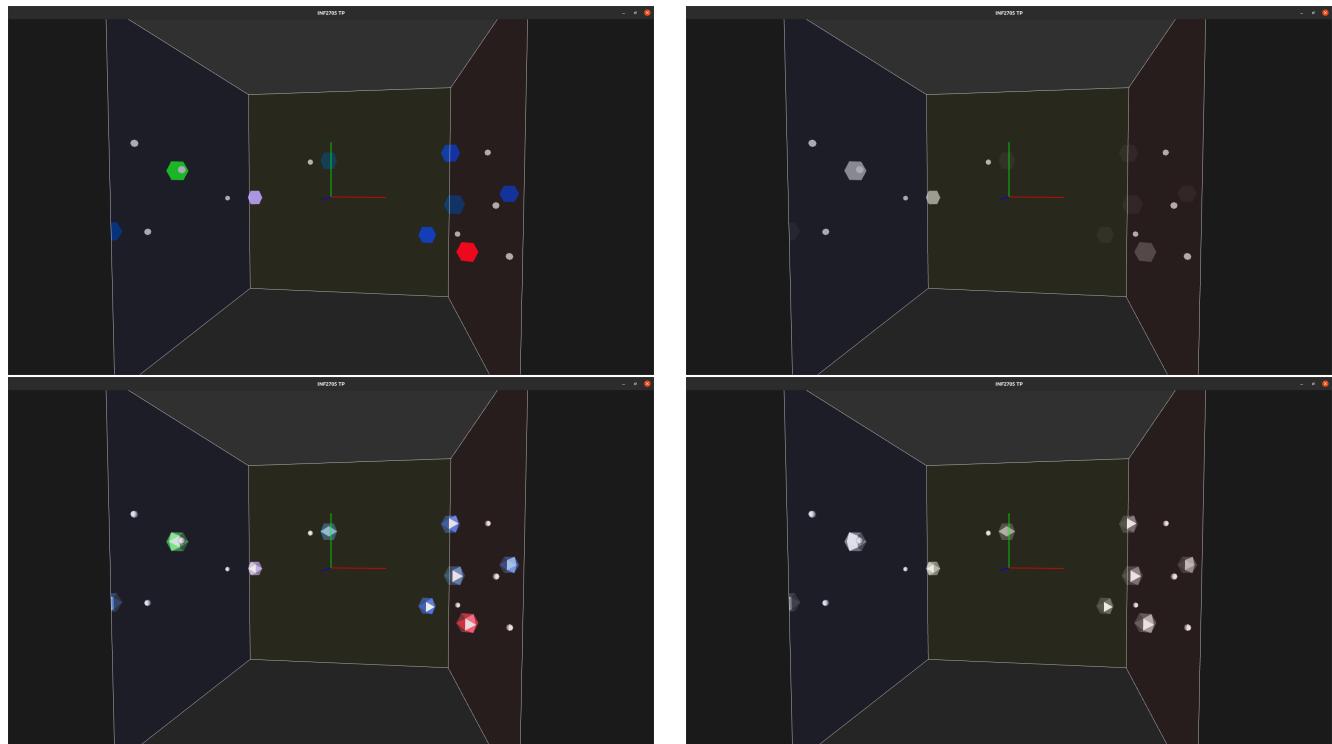


FIGURE 8 – Application de la monochromacité

4. Vous pouvez lire davantage sur le sujet. Plusieurs sources de documentation existe également. Voici un lien qui résume quelques notions, si vous désirez aller plus loin. <https://www.opticiensparconviction.fr/le-daltonisme>

3 Exigences

3.1 Exigences fonctionnelles

Partie 1 :

- E1. Les parois agissent comme des lentilles qui montrent le contenu du système solaire lointain. Un seul appel à `glClear()` est utilisé au début de l'affichage. [7 pts]
- E2. Lorsque les parois du trou de ver sont déplacées (en utilisant les flèches sur le clavier) afin de faire varier la taille du trou de ver, les lentilles continuent d'être bien positionnés sur les parois et les planètes continuent d'être visible à travers. [2 pts]

Partie 2 :

- E3. La caméra peut être positionnée correctement au-dessus des planètes 1 à 9 et on voit les autres planètes ainsi que le trou de ver. [5 pts]
- E4. La caméra est positionnée correctement légèrement derrière le satellite, peu importe la taille de la planète. [1 pt]
- E5. La caméra est positionné soit sur le satellite ou en orbite géosynchrone avec la planète (même rotation que la planète). [1 pt]

Partie 3 :

- E6. Un nuanceur de géométrie est utilisé pour calculer le vecteur normal de chaque triangle et pour donner des valeurs correctes aux variables servant à l'illumination. [4 pts]
- E7. Le nuanceur de fragments calcule correctement l'illumination de Phong en utilisant les valeurs obtenues du nuanceur de géométrie. [3 pts]
- E8. Le nuanceur de fragments consulte la variable uniforme `illumination` pour appliquer ou non l'illumination de Phong. [1 pt]
- E9. (Le logiciel utilise correctement les touches listées à l'annexe A pour faire varier les divers paramètres.)

Partie 4 :

- E10. Un nuanceur est utilisé pour calculer le daltonisme Monochromatique. [2 pts]
- E11. Ce nuanceur calcule correctement l'effet monochromatique avec la formule donnée sur chaque canaux. [1 pt] L'effet de daltonisme Monochromatique est appliqué avec ou sans illumination [1 pt]
- E12. L'effet de daltonisme Monochromatique est appliqué avec ou sans illumination [1 pt]

3.2 Exigences non fonctionnelles

De façon générale, le code que vous ajouterez sera de bonne qualité. Évitez les énoncés superflus (qui montrent que vous ne comprenez pas bien ce que vous faites !), les commentaires erronés ou simplement absents, les mauvaises indentations, etc. [2 pts]

ANNEXES

A Liste des commandes

Touche	Description
q	Quitter l'application
x	Activer/désactiver l'affichage des axes
v	Recharger les fichiers des nuanceurs et recréer le programme
ESPACE	Mettre en pause ou reprendre l'animation
CROCHETGAUCHE	Inverser l'avancement du temps (il recule)
CROCHETDROIT	Remettre l'avancement du temps normalement
g	Permuter l'affichage en fil de fer ou plein
c	Permuter l'affichage des faces arrières
o	Activer l'orbite géosynchrone
DROITE	Augmenter la dimension de la boîte en X
GAUCHE	Diminuer la dimension de la boîte en X
HAUT	Augmenter la dimension de la boîte en Z
BAS	Diminuer la dimension de la boîte en Z
PLUS	Incrémenter la distance de la caméra
MOINS	Décrémenter la distance de la caméra
i	Calculer ou non l'illumination de Phong
m	Appliquer ou non l'effet de daltonisme Monochromatique
0	Afficher le point de vue « normal » (dans notre système solaire)
1	Afficher le point de vue de la planète 1
2	Afficher le point de vue de la planète 2
3	Afficher le point de vue de la planète 3
4	Afficher le point de vue de la planète 4
5	Afficher le point de vue de la planète 5
6	Afficher le point de vue de la planète 6
7	Afficher le point de vue de la planète 7
8	Afficher le point de vue de la planète 8
9	Afficher le point de vue de la planète 9
d	Mode debug : (ne pas utiliser le stencil)
Molette	Changer la distance de la caméra

B Figures supplémentaires



FIGURE 9 – Vous pourrez remplacer le trou de ver par le soleil dans la méthode afficherExosoleil()



FIGURE 10 – Ou même modéliser une sonde qui se déplace vers une planète !

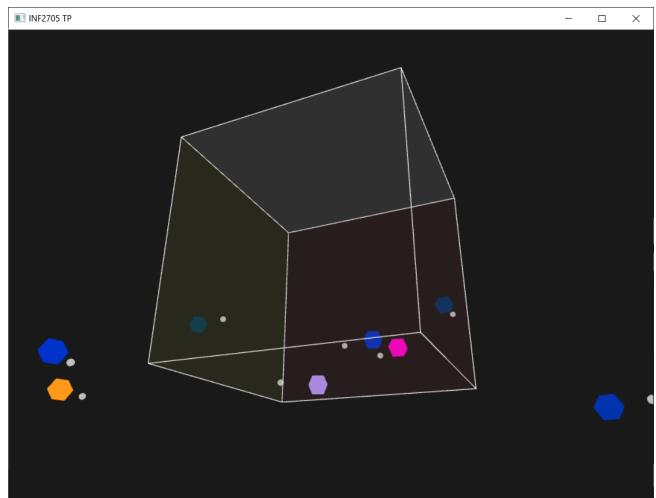
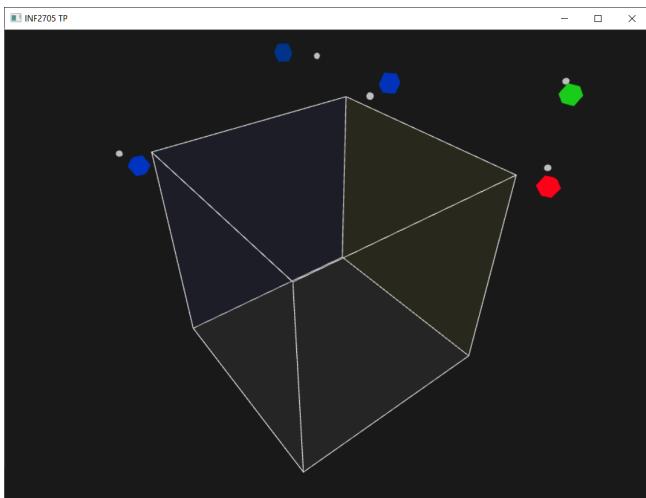


FIGURE 11 – Aucune planète n'est générée en haut ni en bas

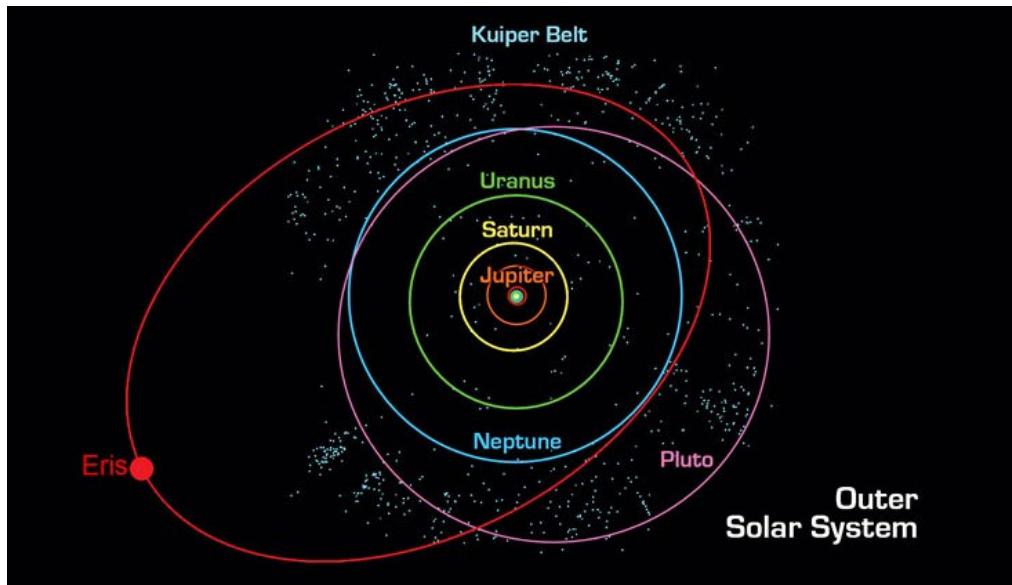


FIGURE 12 – Système solaire externe avec Eris

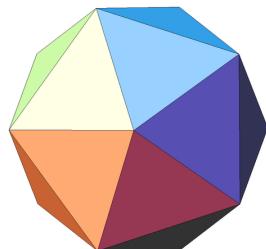


FIGURE 13 – Notre modèle pour les exoplanètes



FIGURE 14 – Le JWST, gracieuseté de la NASA

C Apprentissage supplémentaire

Partie 1 :

1. Utiliser d'autres touches du clavier pour changer la vitesse de déplacement des planètes ou contrôler l'écoulement du temps.
2. Utiliser un plan de coupe à l'intérieur de la scène pour ne pas afficher certaines régions.
3. Utiliser d'autres objets complexe au lieu des planètes, par exemple des théières !.
4. Amener le système solaire à l'intérieur du trou de ver et modifier les parois pour qu'elles agissent comme des miroirs.

Partie 2 :

5. Utiliser la souris pour changer l'orientation du point de vue, tout en restant à la même position au-dessus de la planète/satellite.
6. *[Un peu plus compliqué, mais pas trop]* Utiliser quatre clôtures différentes pour montrer différents points de vue sur la scène. (Il vaut mieux alors appliquer les matrices de projection et de visualisation dans le nuanceur de géométrie.)

Partie 3 :

7. Faites varier la couleur des planète en fonction de la distance au soleil.
8. *[Un peu plus compliqué...mais très cool]* Déplacer la source lumineuse afin que ce soit le soleil qui illumine la scène.

Partie 4 :

9. Appliquer d'autres filtres de couleur des tels que l'inverse des couleurs.
10. Vous pouvez également être créatifs et appliquer d'autres filtres sur des canaux de couleurs particulières comme le bleu, le rouge ou le vert. En d'autres termes, être aveugle par canaux séparément.