POLYTECHNIQUE MONTRÉAL

LE GÉNIE EN PREMIÈRE CLASSE

INF2705 Infographie

Spécification des requis du système

Travail pratique 1

La bestiole et la théière

Département de génie informatique et génie logiciel
École polytechnique de Montréal
Hiver 2017

Table des matières

1 Introduction

Ce document décrit les exigences fonctionnelles et non fonctionnelles du TP1 « *La bestiole et la théière* » du cours INF2705 Infographie.

1.1 But

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

1.2 Portée

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

1.3 Remise

Vous remettrez un fichier `zip` contenant tout le code source du TP et le rapport (`*.cpp`, `*.h`, `*.glsl`, `makefile`, `*.txt`).

Faites « `make remise` » pour créer l'archive « `remise.zip` ».
Vous déposerez ensuite ce fichier dans Moodle.

2 Description globale

2.1 But

Le but de TP est de permettre à l'étudiant de mettre en pratique les fonctions de contrôle du pipeline graphique d'OpenGL pour la modification des matrices et la manipulation de la caméra synthétique : `Rotate()`, `Translate()`, `Scale()`, `PushMatrix()`, `PopMatrix()` et `LookAt()`.

Ce travail pratique lui permettra aussi d'utiliser les fonctions liées aux *Vertex Buffer Objects (VBOs)* : `glGenBuffers()`, `glBindBuffers()`, `glBufferData()` et `glDrawElements()`.

2.2 Travail demandé

Partie 1 : la bestiole

On demande de réaliser un programme permettant d'afficher une bestiole un peu bizarre (probablement d'origine extraterrestre !) avec une tête sphérique de rayon fixe, un corps cubique de taille variable (`tailleCorps`) et quatre pattes de taille fixe (`longPatte` x `largPatte`) formées par des cubes étirés. Cette bestiole pourra aussi transformer son corps en LA théière bien connue en infographie ! La Figure 1 montre cette bestiole sous ses deux formes : cube et théière.

La sphère et les cubes sont tracés par des appels aux fonctions fournies (sans modifier ces fonctions). La tête de la bestiole est une sphère positionnée au milieu de l'arête supérieure (en X+ et Z+). Chaque patte est composé d'un cube étiré et articulé selon un angle (`anglePatte`) (Figure 2). Enfin, la bestiole peut aussi tourner sur elle-même (`angleBestiole`) et se déplacer (`positionBestiole`) dans l'espace de la boîte (Figure 3). Les valeurs de toutes les variables sont contrôlés interactivement.

Ce TP utilise OpenGL 4.x et certains appels à OpenGL sont différents des appels vus au TP0. Consultez les exemples de l'annexe D.

Partie 2 : la caméra synthétique et utilisation de *Vertex Buffer Objects (VBOs)*

Le logiciel permettra de manipuler la caméra synthétique en utilisant a) soit la fonction `LookAt()` pour la placer dans l'espace, b) soit une combinaison de `Translate()` et de `Rotate()`. L'affichage doit être le même, peu importe si on utilise l'une ou l'autre version pour définir le point de vue. La souris contrôlera les deux angles définissant la position de la caméra, tel qu'illustré à la Figure 4.

Enfin, la théière sera affichée en utilisant deux VBOs (sommets et indices) avec les deux tableaux du fichier inclus « `teapot_data.h` ». Ces VBOs doivent être créés au lancement et utilisés ensuite lors de l'affichage.

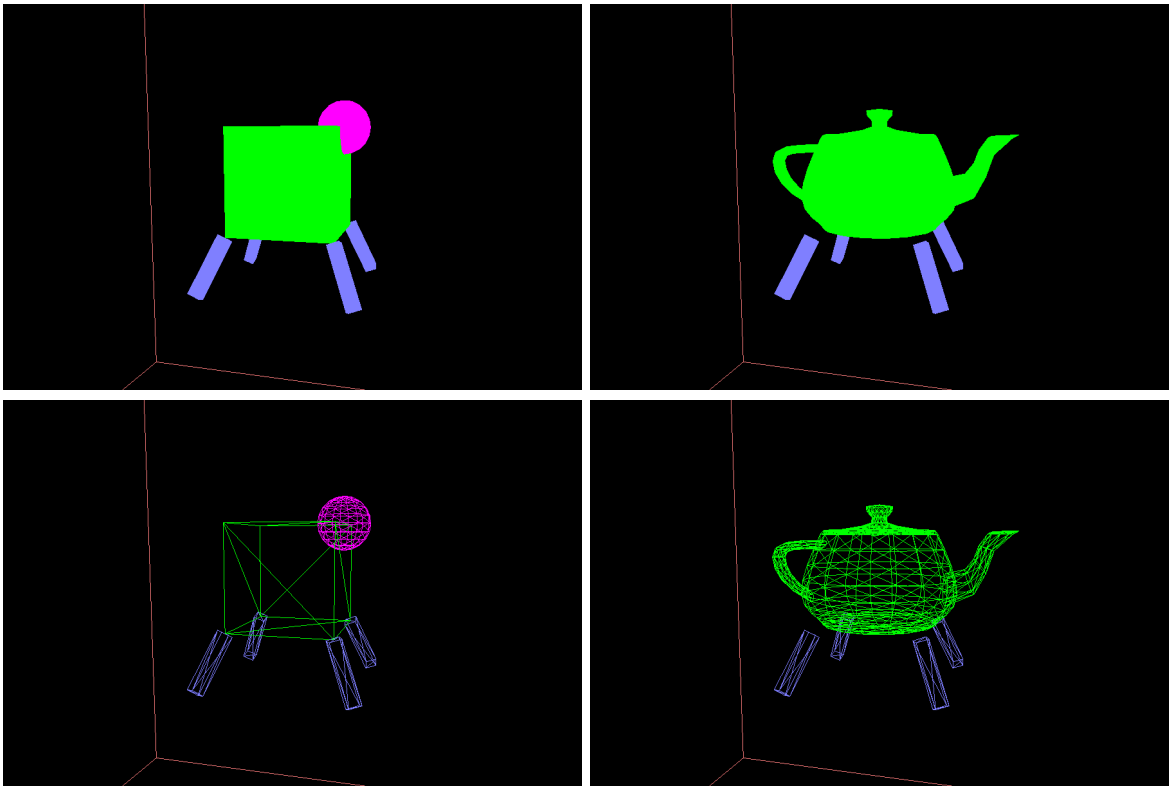

FIGURE 1 – Bestiole sous la forme d'un cube ou d'une théière

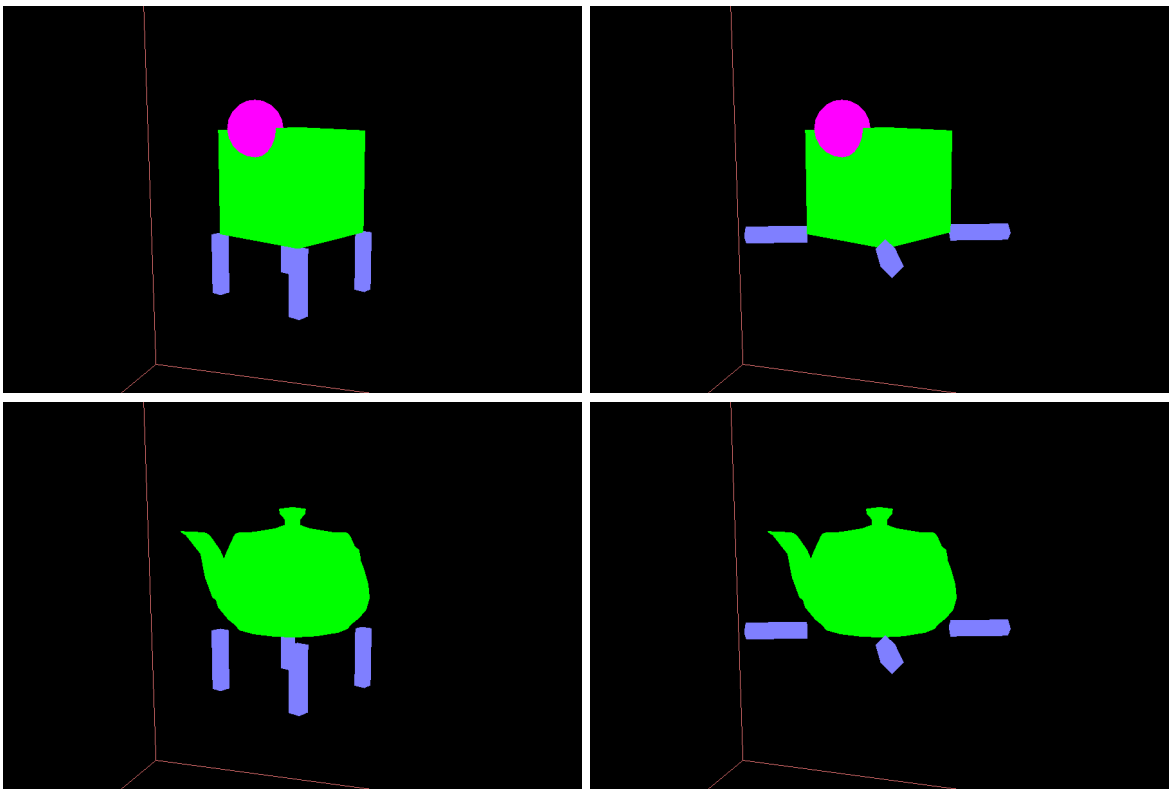

FIGURE 2 – Articulation des pattes

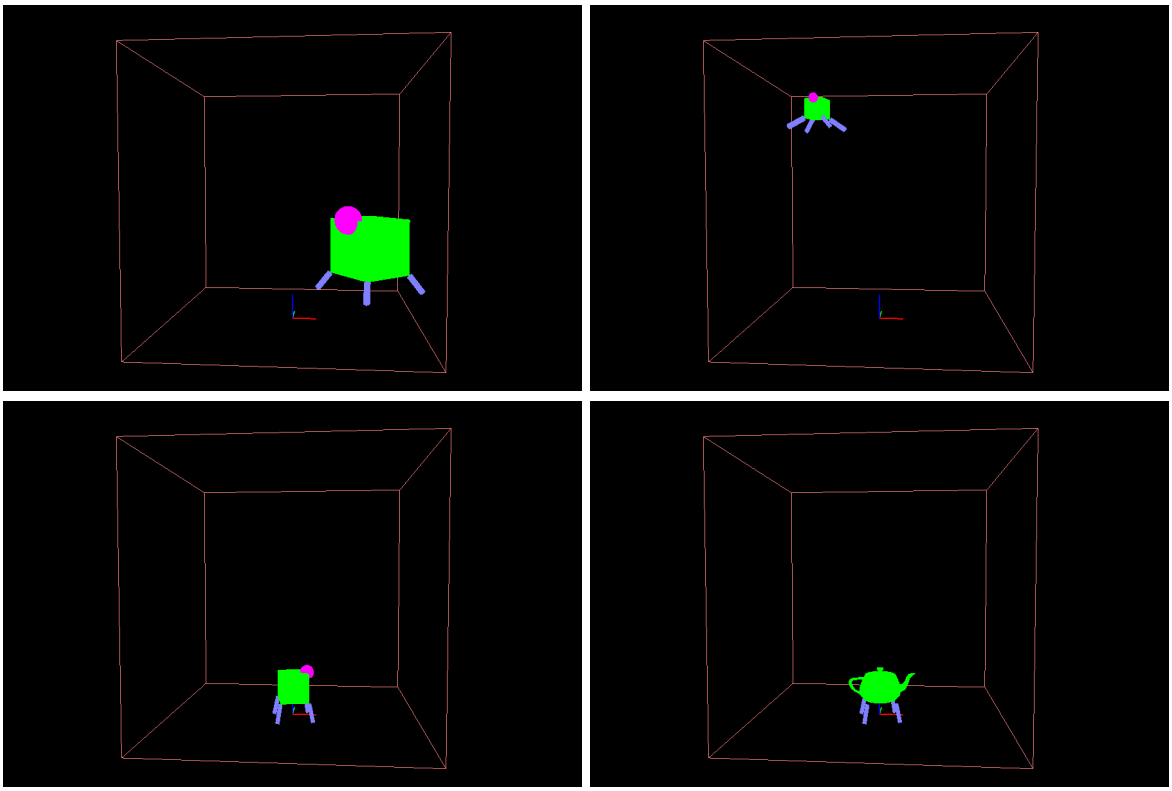

FIGURE 3 – Déplacement, rotation et articulation des pattes en mode animation

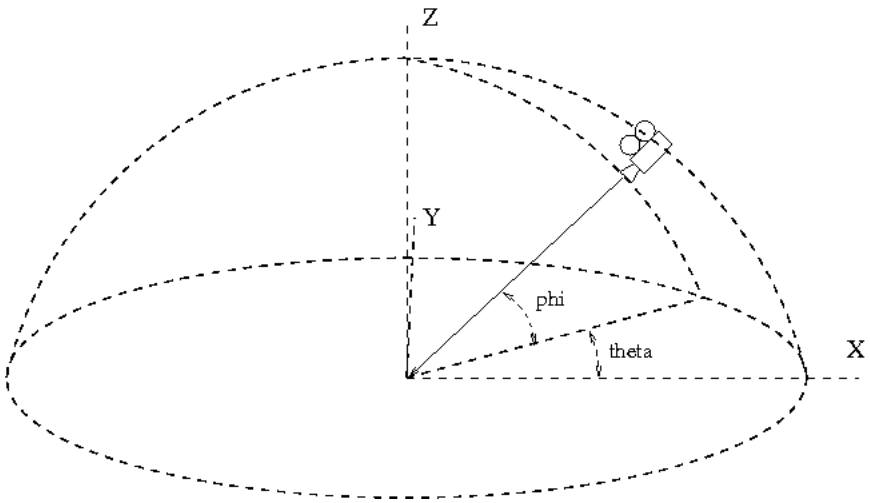

FIGURE 4 – Angles de déplacement de la caméra : *thetaCam* et *phiCam*

3 Exigences

3.1 Exigences fonctionnelles

Partie 1 :

E1. Le corps, la tête et la pattes sont dessinés en utilisant les fonctions `afficherCube()` et `afficherSphere()`.

E2. Les fonctions `Rotate()`, `Translate()` et `Scale()` sont correctement utilisées pour les transformations géométriques nécessaires au dessin de chaque partie de la bestiole.

E3. Les fonctions `PushMatrix()` et `PopMatrix()` sont correctement utilisées pour sauvegarder l'état des matrices pour le dessin de chaque patte.

E4. La bestiole est positionnée selon `positionBestiole` et la taille de son corps est donnée par `tailleBestiole`.

E5. La tête de la bestiole est bien positionnée au milieu de l'arête.

E6. La rotation du corps de la bestiole suit `angleBestiole`.

E7. Les pattes de la bestiole sont attachées aux coins du cube et sont de taille `largPatte` x `largPatte` x `longPatte`.

Partie 2 :

E8. Les fonctions `glGenBuffers()`, `glBindBuffers()`, `glBufferData()` et `glDrawElements()` sont correctement utilisées afin d'utiliser deux VBOs (sommets et indices) pour afficher la théière.

E9. Le corps de la bestiole peut être affiché en utilisant cette théière tel qu'illustré à la Figure 1.

E10. On peut remplacer l'utilisation de `LookAt()`, en gardant le même point de vue, par une série de transformations utilisant `Translate()` et `Rotate()`.

E11. (Le logiciel utilise correctement les touches listées à l'annexe A.)

3.2 Exigences non fonctionnelles

Pour la partie 1, des modifications sont principalement à faire dans la fonction `afficherBestiole()`. Pour la partie 2, des modifications sont principalement à faire dans les fonctions `initiliaser()`, `afficherTheiere()` et `definirCamera()`.

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
`i`	Réinitiliaser le point de vue
`l`	Basculer l'utilisation de LookAt ou de Translate+Rotate pour placer la caméra
`g`	Permuter l'affichage en fil de fer ou plein
`m`	Choisir le modèle affiché : cube, théière
`MOINS`	Reculer la caméra
`PLUS`	Avancer la caméra
`DROITE`	Déplacer la bestiole vers +X
`GAUCHE`	Déplacer la bestiole vers -X
`PAGEPREC`	Déplacer la bestiole vers +Y
`PAGESUIV`	Déplacer la bestiole vers -Y
`BAS`	Déplacer la bestiole vers +Z
`HAUT`	Déplacer la bestiole vers -Z
`FIN`	Diminuer la taille du corps
`DEBUT`	Augmenter la taille du corps
`VIRGULE`	Tourner la bestiole dans le sens anti-horaire
`POINT`	Tourner la bestiole dans le sens horaire
`CROCHETGAUCHE`	Diminuer l'angle des pattes
`CROCHETDROIT`	Augmenter l'angle des pattes
`b`	Incrémenter la dimension de la boite
`h`	Décrémenter la dimension de la boite
`ESPACE`	Mettre en pause ou reprendre l'animation
`BOUTON GAUCHE`	Déplacer (modifier angles) la caméra

B Figures supplémentaires

Allez voir la théière bien connue en infographie sur Internet :
`http://www.sjbaker.org/wiki/?title=The_History_of_The_Teapot`
`http://en.wikipedia.org/wiki/Utah_teapot`.

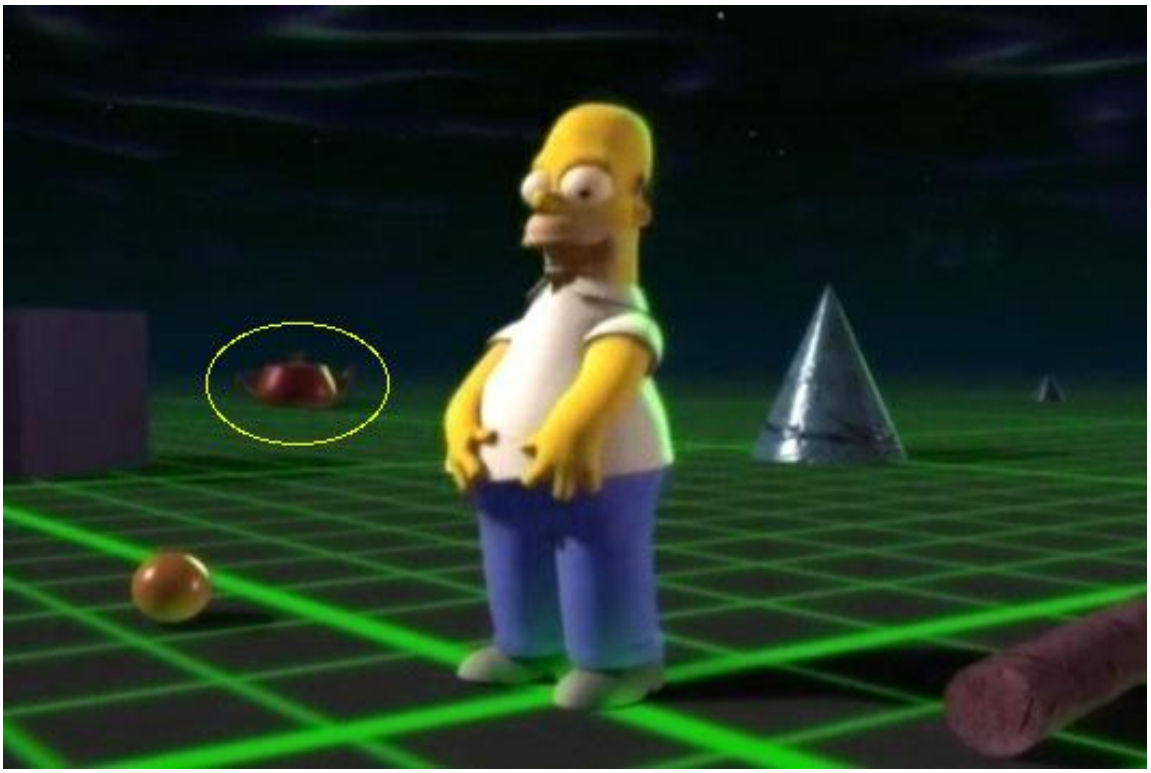

FIGURE 5 – La théière utilisée dans l'épisode *Treehouse of Horror VI*

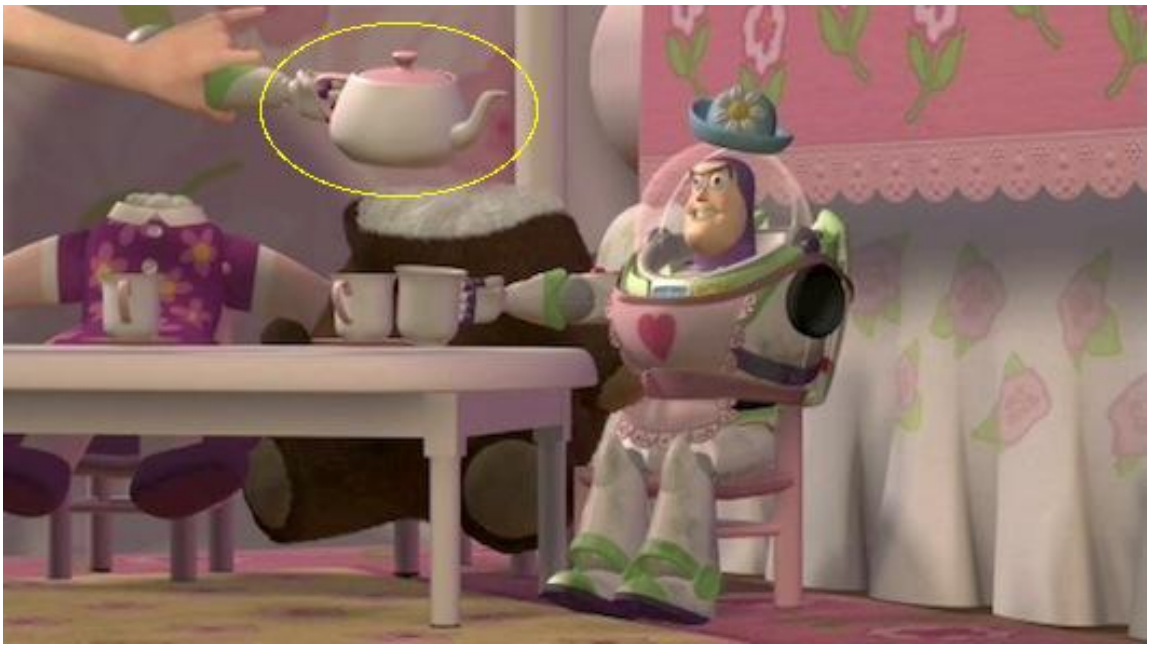

FIGURE 6 – La théière utilisée dans *Toy Story*

FIGURE 7 – La théière utilisée dans un écran de veille (Windows)

C Apprentissage supplémentaire

1. Quel est le nombre minimal de `PushMatrix()`/`PopMatrix()` à utiliser ? Pourquoi faut-il éviter d'en ajouter inutilement ?
2. Allonger les pattes selon la taille du corps.
3. Ajouter des ailes à la bestiole et faites-la voler automatiquement.
4. Utiliser un octaèdre régulier au lieu d'un cube.
5. Utiliser une sphère au lieu d'un cube.

D Utilisation de OpenGL 4.x

Le TP1 utilise OpenGL 4.x et certains énoncés sont différents d'OpenGL 2.x que vous avez utilisé au TP0. En classe, nous avons parlé de l'utilisation de la librairie `glm` et de `std::stack` (utilisés dans la classe `MatricePipeline` du fichier `inf2705.h`). Allez explorer cette classe afin de bien comprendre son fonctionnement.

Voici aussi quelques exemples pour ce premier TP.

— Pour spécifier les transformations élémentaires, on utilisera par exemple :
`matrModel.Translate( 1.0, 0.0, 0.0 );` // translation de (1,0,0)
`matrModel.Rotate( 30, 1.0, 0.0, 0.0 );` // rotation de 30 degrés autour de l'axe des X
`matrModel.Scale( 2.0, 2.0, 2.0 );` // mise à l'échelle par un facteur de 2.0

— Pour gérer la pile sauvegarde de la matrice de modélisation, on utilisera :
`matrModel.PushMatrix();`
`matrModel.PopMatrix();`

— Enfin, avant de tracer, on doit toujours informer la carte graphique des changements faits à la matrice de modélisation en utilisant :
`glUniformMatrix4fv( locmatrModel, 1, GL_FALSE, matrModel );`

On pourra ensuite tracer quelque chose en utilisant cette matrice de modélisation courante :
`afficherCube();`

— Enfin, pour spécifier la couleur d'un objet, l'énoncé :
`glColor3f( 0.0, 1.0, 0.0 );` // vert
est remplacé par :
`glVertexAttrib3f( locColor, 0.0, 1.0, 0.0 );` // vert