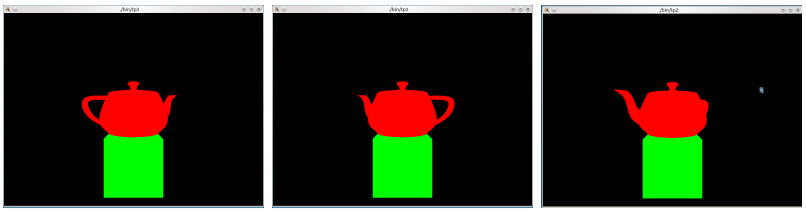
TPs L2 Informatique AII3D : découverte de la synthèse d'images

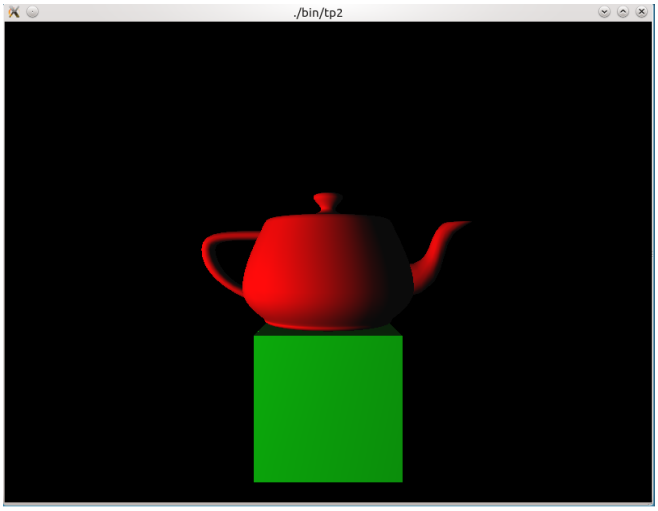
Matériaux et lumières en OpenGL

Ce TP a pour but :

- De comprendre comment mieux structurer un programme de visualistion d'objets avec OpenGL. À la fin de cette première étape, vous pourrez faire tourner la théière sur le cube :



- D'apprendre à affecter des matériaux aux objets et à ajouter/contrôler des sources de lumière pour éclairer la scène.



1 Rappels

L'éclairage en OpenGL

- L'éclairage est activé grâce à la fontion *glEnable(GL_LIGHTING)*. Il peut ensuite être désactivé avec *glDisable(GL_LIGHTING)*.

  A Noter : Toute fonctionnalité activée avec la fonction *glEnable(...)* est désactivée avec la fonction *glDisable(...)*.

- Il y a au moins 8 lumières disponibles en OpenGL, définies par les constantes *LIGHT_0*, *LIGHT_1*, *LIGHT_2*, ... L'activation de ces lumières se fait avec *glEnable(GL_LIGHT0)*, *glEnable(GL_LIGHT1)*, ...Le nombre maximum de lumières est défini par la constante GL_MAX_LIGHT, et il est limité par le système.

- La position des lumières est déterminée de la même façon que pour les primitives ou les objets : avec la matrice modelview. Lorsque vous définissez la position de la lumière, elle est multipliée par la matrice modelview courante. Or la position par défaut de la lumière est définie à l'initialisation et plus du tout après. Donc s'il y a une modification de matrice après la définition de la position de la lumière, cette dernière n'est pas affectée, tout comme une primitive.

  Il faut donc redéfinir la position de la lumière après les modifications de matrice, comme pour tout objet, si vous voulez avoir une lumière stationnaire. Ca a l'air bizarre et compliqué, mais en fait c'est plus pratique : vous pouvez effectuer des transformations aux sources de lumière comme vous voulez comme si elles étaient des objets normaux, et si vous voulez une source de lumière relative au point de vue (comme par exemple une lampe de casque de mineur), il suffit de la définir avant toute modification de matrice.

- Il existe plusieurs types de lumières en OpenGL :

  Ambiante : Lumière générale, vient de toutes les directions et repart dans toutes les directions.

  Diffuse : Lumière venant d'une direction particulière. Elle est par conséquent modifiée suivant l'angle de réflexion de l'objet rencontré. La reflexion est par contre uniforme dans toutes les directions.

  Spéculaire : Lumière venant d'une direction et repartant dans une direction particulière (effet miroir).

  Par exemple, la lumière ambiante est la couleur de l'objet sans éclairage, la lumière diffuse est la couleur de l'objet illuminé par une lumière ambiante (lumière non directe) et la couleur spéculaire est la couleur donnée par un éclairage direct, par exemple des métaux ont en général une lumière spéculaire blanche (tache blanche sur une sphère en métal par exemple). On peut également caractériser la lumière d'émission.

- Changer les caractéristiques d'une lumière se fait grâce à la fonction *glLightfv(source, caractéristiques, couleur).*

  Le paramètre *source* sert à indiquer la source de lumière concernée (GL_LIGHTX).

  Le paramètre *caractéristiques* permet d'indiquer quelle caractéristique de la source va être modifiée :
  GL_AMBIENT : couleur de la lumière ambiante ajoutée à la scène (valeur par défaut : (0.0,0.0,0.0,1.0)).
  GL_DIFFUSE : couleur de la lumière diffuse liée à la source lumineuse (valeur par défaut : (1.0,1.0,1.0,1.0) pour *GL_LIGHT0* et (0.0,0.0,0.0,1.0) pour les autres).
  GL_SPECULAR : couleur de la lumière spéculaire liée à la source lumineuse (valeur par défaut : (1.0,1.0,1.0,1.0) pour *GL_LIGHT0* et (0.0,0.0,0.0,1.0) pour les autres).
  GL_POSITION : position de la lumière (x,y,z,w). Si w = 0, la lumière est placée à l'infini.
  GL_SPOT_DIRECTION : orientation du cone d'ouverture dans le cas d'un spot.
  ... (Il en existe d'autres).

  Le paramètre *couleur* correspond à la couleur en RVBA, sous forme d'un vecteur (tableau).

Les matériaux en OpenGL

- Pour pouvoir gérer correctement les lumières il faut d'abord définir pour CHAQUE objet ses caractéristiques de luminosité. Il faut donc définir de quelle façon le matériau de l'objet va intéragir avec la lumière.

- Changer le matériau d'un objet se fait grâce à la fonction *glMaterialfv(faces,caractéristiques,couleur).*

  Le paramètre *faces* correspond aux faces concernées :
  GL_FRONT (faces de devant)
  GL_BACK (faces arrières)
  GL_FRONT_AND_BACK.

  Le paramètre *caractéristiques* correspond aux caractéristiques de luminosité de l'objet. À savoir :
  GL_AMBIENT (valeur par défaut : (0.2,0.2,0.2,1.0))
  GL_DIFFUSE (valeur par défaut : (0.8,0.8,0.8,1.0))
  GL_AMBIENT_AND_DIFFUSE
  GL_SPECULAR (valeur par défaut : (0.0,0.0,0.0,1.0))
  GL_SHININESS : focalisation spéculaire (valeurs : 0 -¿ 128)
  GL_EMISSION : couleur émise (valeur par défaut : (0.0,0.0,0.0,1.0))
  GL_COLOR_INDEXES : indexes des couleurs ambiante, diffuse et spéculaire.

  Le paramètre *couleur* correspond à la couleur en RVBA, sous forme d'un vecteur (tableau).

2 Modifier la structure du Programme

- Telechargez les fichiers *objet.h* et *objet.c* disponibles aux adresses suivantes :
  www.irit.fr/ Loic.Barthe/Enseignements/TPs_OpenGL/M1_IO5/TP1/objet.h
  www.irit.fr/ Loic.Barthe/Enseignements/TPs_OpenGL/M1_IO5/TP1/objet.c
  et positionnez les dans le répertoire contenant votre fichier *main.c*

- Dans le fichier main.c, ajoutez une variable globale *pObjet _obj ;*

- Au début du main créez le pointeur avec la ligne *_obj = creerObjet() ;*

- Dans la fonction *void display (void)* ajoutez la fonction *afficheObjetOpenGL (_obj) ;* juste après l'initialisation de la pile de matrice GL_MODELVIEW.

- Dans la fonction void *afficheObjetOpenGL (pObjet obj)* du fichier *objet.c* recopiez les quelques lignes traçant le cube et la teapot et enlevez les de la fonction *display* du fichier *main.c.*

- Dans le fichier *Makefile.common* ajoutez *objet.c* à la variable *APPFILES* (le séparateur est un espace).

  Attention : à chaque fois que vous créez un nouveau fichier .c il vous faut l'ajouter dans le fichier *Makefile.common* pour qu'il soit correctement compilé.

- Compilez le programme et executez le. Si tout va bien, vous devez avoir le même résultat qu'avant ces modifications.

  Présentez la nouvelle structure du programme et expliquez l'intérêt de procéder ainsi.

- Expliquez brièvement le principe d'empilement et de dépilement de matrices.
  L'empilement est effectué avec *glPushMatrix()* et le dépilement avec la fonction *glPopMatrix().*

  Pour finir les modifications précédentes, la fonction void *afficheObjetOpenGL (pObjet obj)* du fichier *objet.c* doit commencer par la fonction *glPushMatrix()* et finir par la fonction *glPopMatrix().* Faites le dans votre code et expliquez pourquoi. Utilisez également ces fonctions autour du positionnement/tracé de la teapot.

- *Évènement clavier :* afin de se familiariser avec cette nouvelle structure de code, vous aller coder le mouvement de la théière autour de l'axe y. La touche 'a' permet d'activer ou de désactiver la rotation.

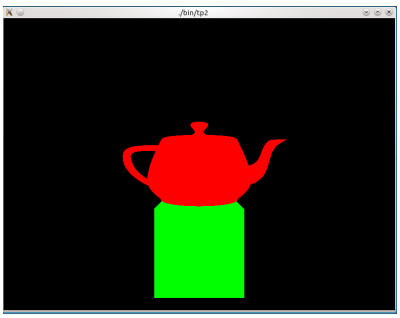
  La glut permet d'appeler une fonction de façon répétitive à tout moment, quand aucun évènement n'est détecté. Cette fonction que l'on appellera *noEvent* est activée avec la glut au même niveau que la fonction de rendu (*glutDisplayFunc(display)*) et la fonction de gestion des évènements clavier (*glutKeyboardFunc(key)*) en utilisant la fonction *glutIdleFunc(noEvent)* dans la fonction *main.*

  Comme les fonctions *display* et *key*, cette fonction ce déclare dans votre fichier *main.c* : *void noEvent (void).* Elle utilise la fonction *void tourneObjet (pObjet obj, float pasRotation)* du fichier *objet.c* pour modifier l'attribut *theta* de l'objet. La rotation d'angle *obj→theta* est ensuite effectuée dans la fonction *void afficheObjetOpenGL (pObjet obj).*

3 Éclairages et Matériaux

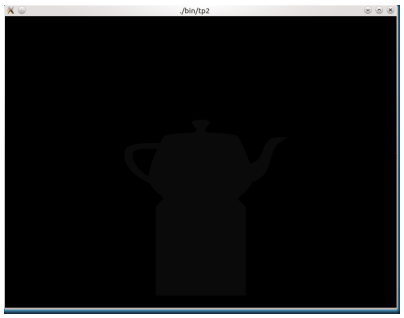
Éclairer la scène

- Suite au TP1, votre scène devrait resembler à peu près à :

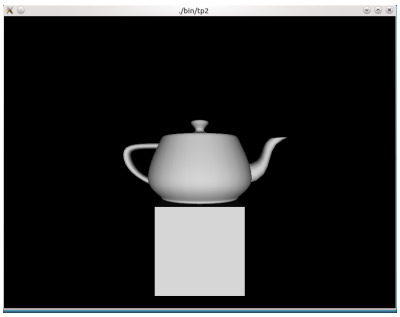


- Dans un premier temps, vous allez éclairer vos objets. Ajoutez la commande *glEnable(GL_NORMALIZE)* juste après l'activation du Z-Buffer. Cette commande permet de normer automatiquement les normales des différents objets afin que leur éclairage soit calculé correctement.

- L'éclairage est activé à l'aide de la fonction *glEnable(GL_LIGHTING)*.
  Que se passe-t-il quand vous activez l'éclairage ? Pourquoi ?



- Gestion des évènements : utilisez la touche e pour activer/désactiver l'éclairage. Utilisez également la touche 0 pour éclairer/éteindre la lumière *LIGHT_0*.
  Que se passe-t-il quand elle est éclairée ? Pourquoi ?

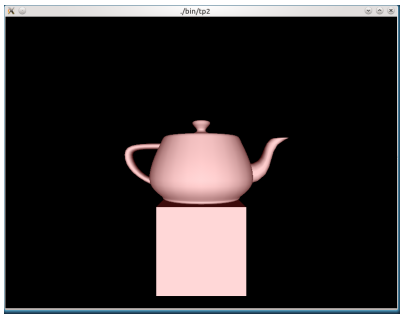


Changer l'éclairage

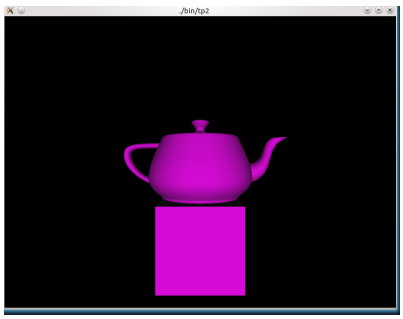
- Pour ce TP, vous pourrez utiliser le type Vector4 défini dans les fichiers *vector4.h* et *vector4.c* afin de vous faciliter la manipulation des couleurs des lumières et des matériaux. Vous pouvez les récupérer à l'adresse suivante :

  www.irit.fr/ Loic.Barthe/Enseignements/TPs_OpenGL/M1_IO5/TP2/vector4.h
  www.irit.fr/ Loic.Barthe/Enseignements/TPs_OpenGL/M1_IO5/TP2/vector4.c

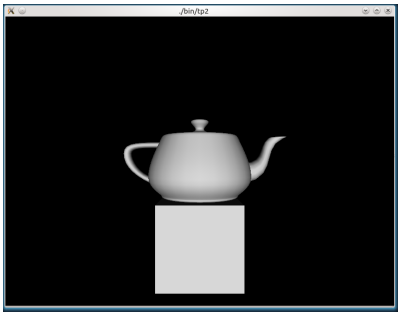
- A l'aide de la variable *float light0_ambient []= 0.0,0.0,0.0,1.0* (ou *Vector4 light0_ambient* initialisée avec la fonction *setVector4 (...)*) et de la fonction *glLightfv(GL_LIGHT0, GL_AMBIENT, light0_ambient)* modifiez la couleur de la lumière ambiante émise par la lampe0. Observez et commentez le résultat.



- De la même façon, observez et commentez ce qui se passe quand on change la composante diffuse avec la fonction *glLightfv(GL_LIGHT0, GL_DIFFUSE, light0_diffuse)*.

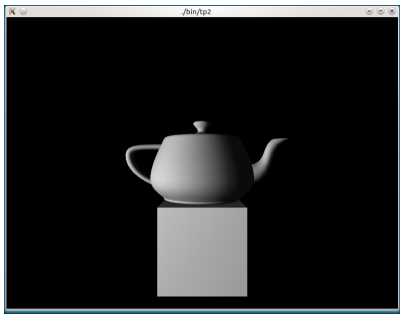


- Toujours pareil, observez et commentez ce qui se passe quand on change la composante spéculaire avec la fonction *glLightfv(GL_LIGHT0, GL_SPECULAR, light0_specular)*.



- Vous pouvez déplacer la source de lumière avec la fonction *glLightfv(GL_LIGHT0, GL_POSITION, light0_position)*.
  Attention, la lampe se positionne dans le repère objet (dans la fonction display, dans la pile de matrice *GL_MODELVIEW)* et toutes les lampes doivent être positionnées avant le tracer des objets.

  Placez la lampe 0 dans une position fixe, sur la gauche de la teapot.



Les matériaux

- Avant de regarder le matériau de l'objet, nous allons prendre un éclairage blanc commun :
  light0_ambient []= 0.01,0.01,0.01,1.0 ;
  light0_diffuse []= 1.,1.,1.,1.0 ;
  light0_specular []= 1.,1.,1.,1.0 ;
  light0_position []= -6.,0.,0.,1.0 ;

- Nous allons maintenant modifier les matériaux. Pour simplifier, nous allons utiliser le même matériau pour le cube et la teapot (même si les couleurs pourront différer par la suite).

  Note : Le matériau est un attribut des objets qui sont affichés. On peut imaginer que l'on peut utiliser un matériau différent pour le cube et la teapot. Ainsi, le matériau se définit dans notre cas dans la fonction d'affichage de l'objet (dans le fichier *objet.c*), avant le tracé.

- A l'aide de la variable *float obj_ambient [] = 0.2,0.2,0.2,1.0* et de la fonction *glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT, obj_ambient)* modifiez la composante ambiante du matériau. Observez et commentez le résultat.

- De la même façon, observez et commentez ce qu'il se passe quand on change la composante diffuse avec la fonction *glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE, obj_diffuse)*.

- Toujours pareil, observez et commentez ce qui se passe quand on change la composante spéculaire avec la fonction *glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, obj_specular)*.

- Enfin, observez et commentez ce qui se passe quand on change la composante "brillance" avec la fonction *glMaterialf(GL_FRONT_AND_BACK, GL_SHININESS, obj_shininess)*.
  Dans quel intervalle doit-on prendre cette valeur ?

- À vous maintenant de jouer sur les paramètres des matériaux pour créer différents types d'objets (plâtre, miroir, métal, PVC, ...)