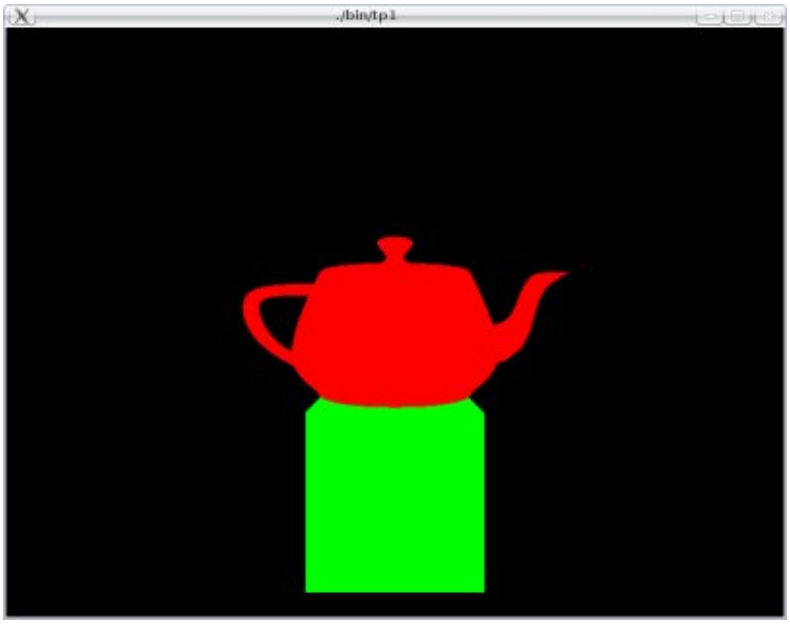
Création d'un viewer d'objets 3D : découverte de la librairie graphique OpenGL.

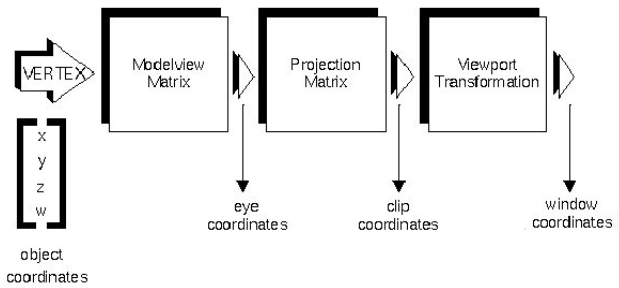
Le but de ce TP est de se familiariser avec la librairie et le mode d'affichage OpenGL, le positionnement d'un objet dans la scène, la gestion de la couleur en RVB ainsi que l'animation d'un objet. En même temps, il va être possible de se familiariser avec la programmation évènementielle, l'élimination des parties cachées d'un objet et le bouble buffer.

A la fin de ce TP vous aurez créé et animé la scène suivante :



Rappels : Le processus de Visualisation en OpenGL

- Le rôle d'OpenGL est de **projeter une scène 3D sur une image 2D** : l'écran.
- Pour cela, chaque objet défini dans la scène subit plusieures transformations.



- Quatre transformations utilisées au cours du processus de création d'une image :

  1. Transformation de **modélisation** (Model) :
  Permet de créer la scène à afficher par création, placement et orientation des objets qui la composent.

  2. Transformation de **visualisation** (View) :
  Permet de fixer la position et l'orientation de la caméra.

  3. Transformation de **projection** (Projection) :
  Permet de fixer les caractéristiques optiques de la caméra (type de projection, angle d'ouverture, ...).

  4. Transformation d'**affichage** (Viewport) :
  Permet de fixer la taille et la position de l'image sur la fenêtre d'affichage.

- Chaque transformation est définie par une matrice.
- OpenGL compose ces matrices afin d'appliquer une transformation sur chaque objet de la scène.
- Le résultat de cette transformation est l'objet dans les coordonnées de l'image à afficher.
- OpenGL fourni un ensemble de fonctions permettant de manipuler les matrices de transformation.
- On choisi la matrice courante à manipuler avec *glMatrixMode*.
- Ensuite on utilise une fonction permettant de définir ou de modifier la matrice courante : *glLoadIdentity, glLoadMatrix, glMultMatrix, glTranslate, glRotate, glScale, gluLookAt, etc...*

Introduction

Avant de pouvoir commencer votre TP, il vous faut installer la structure de programme de départ. Suivez les instructions suivantes :

- Créez un répertoire qui sera votre répertoire de travail pour les TPs d'OpenGL
- Téléchargez dans ce répertoire le fichier suivant :
  http ://www.irit.fr/ Loic.Barthe/Enseignements/TPs_OpenGL/TP1/intro_OpenGL.tar.gz
- Décompressez l'archive dans le répertoire puis effacez la
- Dans un shell, placez vous dans le répertoire où vous avez décompressé les fichiers et compilez le programme en tapant : *make*
  Les fichiers objets (.o) et l'exécutable (nommé "tp1") sont dans le répertoire "bin"
- Lancez le programme avec la commande : *./bin/tp1*
- Editez le programme principal "main.c". Il est suggéré d'utiliser l'éditeur "kate" qui se lance avec la commande : *kate main.c &*

Pour pouvoir effectuer les TPs correctement, vous aurez souvent besoin de vous référer au guide de programmation (html) :

$$http : //www.opengl.org/$$

Vous pouvez aussi accéder à l'aide en ligne avec la commande :

$$man < nom\_fonction >$$

Ou encore utiliser konqueror et taper :

$$man :< nom\_fonction >$$

Effacement de la fenêtre

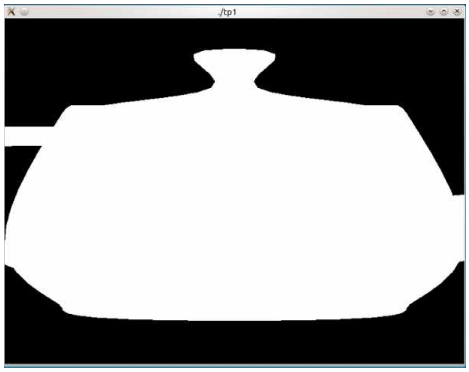
- Que se passe-t-il quand vous exécutez le programme ? Pourquoi ?
- Dans la fonction "display" du fichier "main.c", ajoutez la fonction suivante en première ligne :
  glClear (GL_COLOR_BUFFER_BIT) ;
  Que se passe-t-il ? Pourquoi ?
- Ajoutez juste après la fonction :
  glClearColor (1.0, 1.0, 1.0, 0.0) ;
  Quelle est la couleur (1.0, 1.0, 1.0) ? On notera que la 4ème composante de la couleur est la composante Alpha qui sert à régler la transparence. Pour les TPs, cette composante sera fixée à 0.0 (pas de transparence). Que se passe-t-il ? Pourquoi ?
  Positionnez cette fonction à la bonne place pour que la fenêtre soit effacée avec la couleur souhaitée. Faites effacer la fenêtre en rouge, puis en gris foncé.

L'effacement du buffer image est en général la première chose que l'on fait dans la fonction de rendu (display).

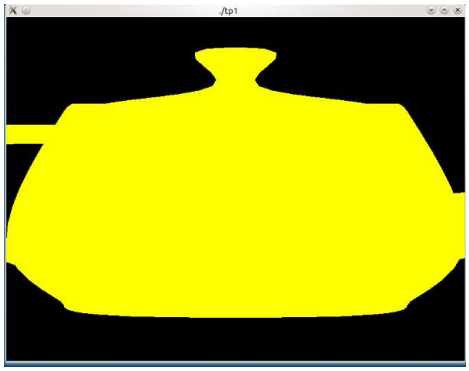
Afficher et positionner la théière

- La fenêtre et la matrice de projection ont des valeurs par défaut, ainsi, il est possible de tracer directement un objet. Tracez une théière (teapot) de rayon 1 centrée sur l'origine du repère scène à l'aide de la fonction suivante :

  glutSolidTeapot(1.) ;



- La fonction *glColor3f* permet de modifier la couleur d'affichage des objets à l'aide de 3 flottants (rouge,vert,bleu) passés en paramètre. Placez cette fonction au mieux pour que la théière soit tracée en jaune.



- Il s'agit maintenant de retrouver les valeurs par défaut de la fenêtre. On précise que la projection par défaut est une projection orthogonale (pas de perspective). Intéressons nous tout d'abord à la fenêtre de projection. Elle définit la position et la taille de l'image qui va être affichée dans la fenêtre de l'application OpenGL. Elle est définie par la fonction :

  glViewport(x,y,l,h) ;

  Que représentent les paramètres x,y,l et h ?
  Insérez cette fonction dans votre programme et retrouvez la valeur par défaut de ses paramètres.

- Nous allons ajouter une matrice de projection perspective. Pour ça, il nous faut ajouter à la fonction de rendu (display) les fonctions permettant de contrôler la projection.

  Dans un premier temps, il nous faut passer en mode de modification de la matrice de projection avec la fonction :

  glMatrixMode (GL_PROJECTION) ;

  Puis on va initialiser la matrice de projection avec la fonction :

  glLoadIdentity () ;

Quelle est la valeur de la matrice de projection ?
Que se passe-t-il au niveau de l'affichage ?

Modifiez maintenant la matrice de projection perspective à l'aide de la fonction suivante :

gluPerspective(60.,800./600.,1.,100.) ;

Que représentent les paramètres de cette fonction ?
Pourquoi ne voit-on plus la teapot à l'écran ?

- Afin de pouvoir voir la teapot, nous allons la déplacer : nous allons utiliser les matrices de transformation (mode MODELVIEW).
  Il nous faut donc dans un premier temps activer le mode de modification de matrice de transformation (ceci désactivera le mode de modification de matrice de projection que nous avions activé précédemment). Ceci est fait en ajoutant au programme la fonction :

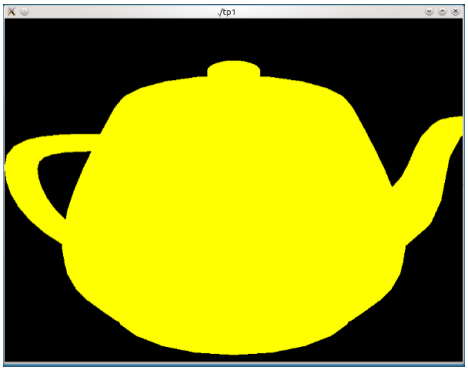
  glMatrixMode (GL_MODELVIEW) ;

  Une fois la fonction précédente ajoutée à votre programme, initialisez la matrice avec la fonction :

  glLoadIdentity () ;

  Déplacez maintenant la théière en faisant la translation suivante :

  glTranslatef(0.,0.,-4.) ;

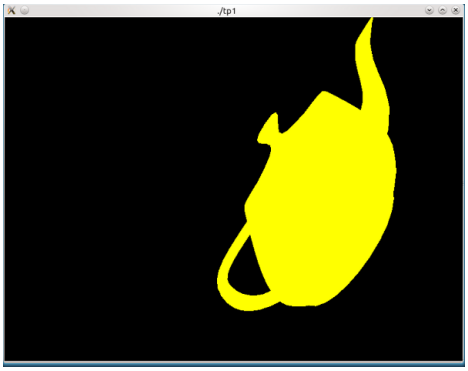


Que s'est-il passé ? Pourquoi ?

- Modifiez la translation, et ajoutez une rotation et une mise à l'échelle à l'aide des fonctions :

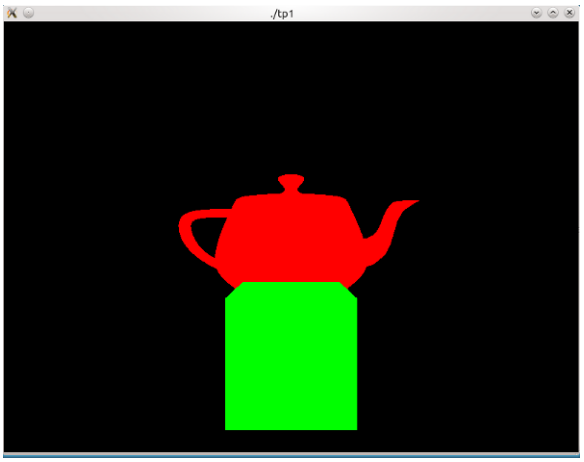
  glRotatef(...)
  glScalef(...)

  pour placer la teapot afin que le rendu s'approche de l'image suivante :



Afficher la scène

- Mettons maintenant les objets dans la scène de la façon suivante :



Les objets (teapot et cube) sont à une profondeur de 5 en z (plus de rotation ni de mise à l'échelle).
Le cube peut être tracé avec la fonction :

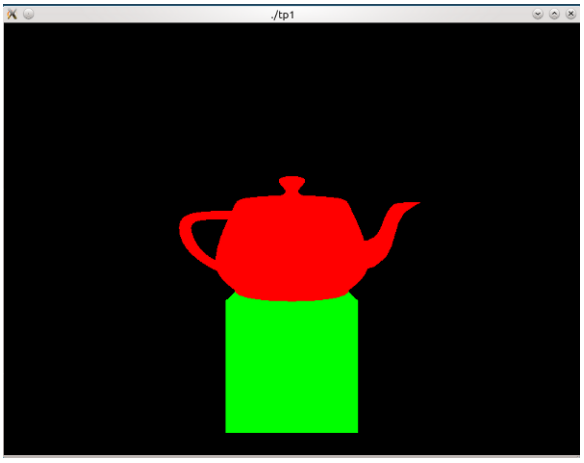
glutSolidCube(...) ;

- Pour l'instant, le cube s'affiche devant la teapot. Pour corriger ça, nous allons maintenant activer le test de profondeur, appelé Z-Buffer, pour éliminer les faces cachées.

  Tout d'abord, il faut initialiser le mode d'affichage avec test des profondeurs dans la fonction *glutInitDisplayMode* en ajoutant le paramètre *GLUT_DEPTH*.

  Ensuite, il faut activer le test des profondeurs à l'aide de la fonction *glEnable(GL_DEPTH_TEST)* placée juste après la création de la fenêtre principale (dans le *main(...)*). Il est à noter que le Z-Buffer peut être désactivé n'importe quand avec la commande *glDisable(GL_DEPTH_TEST)*.

  Enfin, il faut aussi effacer cette mémoire (buffer) au début de chaque tracé, en même temps que la mémoire d'image (color buffer). Ceci est effectué en ajoutant | *GL_DEPTH_BUFFER_BIT* dans la fonction *glClear*.
  La teapot doit maintenant apparaître posée sur le cube.

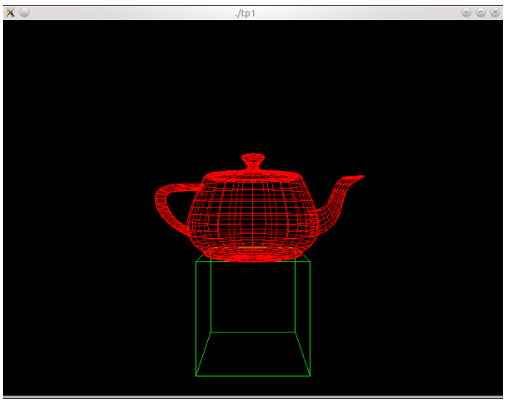


- La qualité de l'affichage peut être améliorée en utilisant le double buffer.
  Le double buffer est activé en remplaçant dans la fonction glutInitDisplayMode la valeur *GLUT_SINGLE* par *GLUT_DOUBLE*. Le "swap" des buffers est alors réalisé à la fin de la fonction de rendu (display) avec la fonction *glutSwapBuffers()* placée juste avant la fonction *glFlush()*.

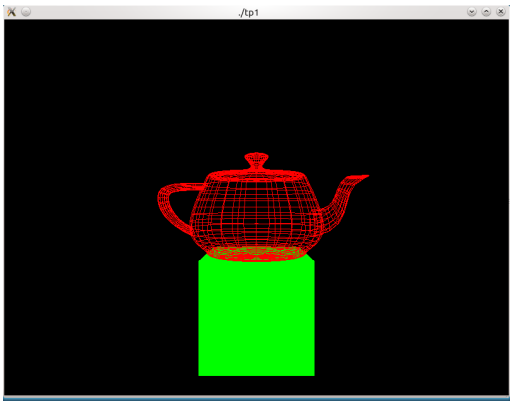
- Gestion des évènements : Une lettre (par exemple, la lettre a) est testée dans un switch par le test *case 'a'*.
  Utilisez la lettre z pour désactiver le Z-Buffer et la lettre Z pour le réactiver.
  Si vous souhaitez forcer le réaffichage de la fenêtre, cela peut être fait avec la fonction *glutPostRedisplay()*.

Modifier l'Affichage

- L'affichage par défaut est un mode plein. On peut passer en affichage filaire en utilisant la fonction :
  glPolygonMode (GL_FRONT_AND_BACK, GL_LINE) ;
  On peut repasser en mode plein avec les paramètres :
  glPolygonMode (GL_FRONT_AND_BACK, GL_FILL) ;
  Modifiez votre programme pour que la teapot et le cube soient affichés en filaire.



  Modifiez à nouveau votre programme pour que seule la teapot soit affichée en filaire.



- Gestion des évènements : Utilisez la lettre w pour passer en mode d'affichage filaire et la lettre W pour repasser en mode d'affichage plein.