PACT

TP OpenGL

28 Novembre 2018
J. Le Feuvre

Le but de ce TP est de comprendre les principes généraux d'OpenGL. Ce TP est effectué en complément du cours d'introduction à la synthèse 3D disponible en ligne.

Prérequis

- Téléchargez le fichier d'exemple ici.
- Décompressez l'archive, ouvrez un terminal et allez dans le repertoire créé
- Ouvrez le fichier main.cpp avec votre éditeur de texte ou de code favori
- complilez le programme en tappant *make*
- rendez le programme executable en tappant *chmod +x gltest*

Le fichier main.cpp est un programme C mettant en place quelques fonctionnalités de base openGL:

- création d'une fenetre pour l'affichage OpenGL
- création de contexte et mise en place d'une configuration par défault openGL
- chargement d'une texture RGB à partir d'un fichier BMP
- compilation de shaders OpenGL

Ces fonctions sont spécifiques au laguage C et aux bibliothèques GLUT et GLEW simplifiant la mise en oeuvre d'OpenGL. Dans le cadre de votre projet, vous aurez certainement à travailler avec d'autres bibliothèques (Android, LWJGL, ou autres). Vous serez bein entendu libre d'utiliser l'outil qui vous semblera le plus approprié pour votre projet.

Lorsque vous modifierez le fichier main.cpp, vous **devrez recompiler** l'executable en tapant *make*.

Le programme comporte déjà quelques exemples:

- regardez l'aide: *./gltest -h*
- afficher un caré: *./gltest*
- afficher un cube: *./gltest -mode 1*
- afficher une teillère: *./gltest -mode 2*

- déplacement simple de caméra pour le cube et la teillère via la souris (cliquez dans la fenêtre et déplacez la souris en gardant appuyé)

Changements de géométrie

Changez le type de primitive graphique dans code display_quad():

- utilisez GL_LINES au lieu de GL_TRIANGLES: qu'observe-t-on?
- utilisez GL_LINE_STRIP au lieu de GL_TRIANGLES: qu'observe-t-on?

Comme vu en cours, OpenGL permet de spécifier chaque vertex dune primitive geometrique. Vous utiliserez la primitive glVertex3f qui donne les coordonnées du prochain vertex dans la primitive.

Etudiez le code de la fonction display_quad (on ignorera la partie vbo) et modifiez-le pour afficher:

- un trapèze rectangle
- un trapèze isocèle
- un triangle
- un pentagone
- un cercle

Vous pouvez utiliser les modes GL_TRIANGLES (3 vertex par triangles), GL_TRIANGLE_FAN (3 vertex pour le premier triangle, puis 1 vertex par triangle, le triangle construit prenant le premier point du premier triangle et le dernier point spécifié pour le triangle précédent et le vertex spécifié) ou GL_TRIANGLE_STRIP (3 vertex pour le premier triangle, puis 1 vertex par triangle, le triangle construit prenant les deux derniers points du triangle précédent et le vertex spécifié).

Etudiez le code de la fonction drawCube (on ignorera la partie vbo) et modifiez-le pour afficher:

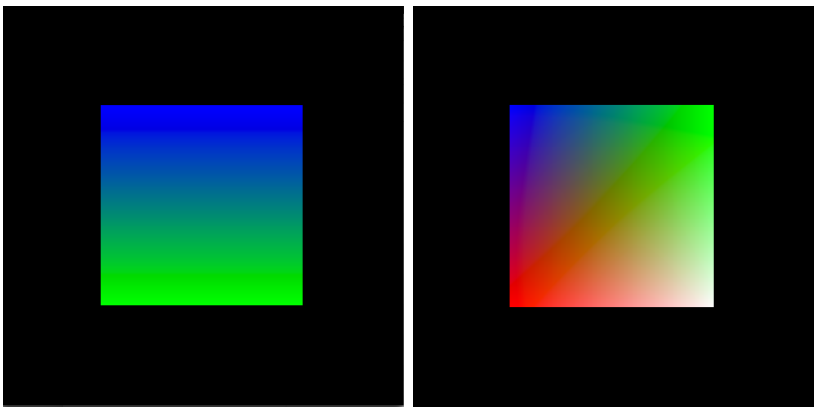
- un cube sans bas ni haut
- une pyramide tronquée carrée
- une pyramide carrée
- une sphère

Il est conseiller de rajouter des modes et des fonctions dédiées plutot que de changer tout le code à chaque fois.

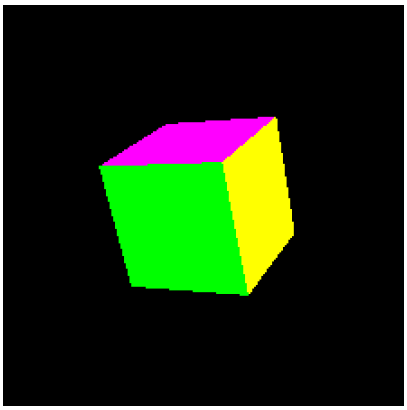
Changements de Couleur

Comme vu en cours, OpenGL permet de spécifier la couleur de chaque vertex dans une primitive geometrique. Vous utiliserez la primitive glColor3f qui assigne la couleur courante pour les prochains vertex.

Modifiez la fonction display_quad pour déssiner la figure suivante:



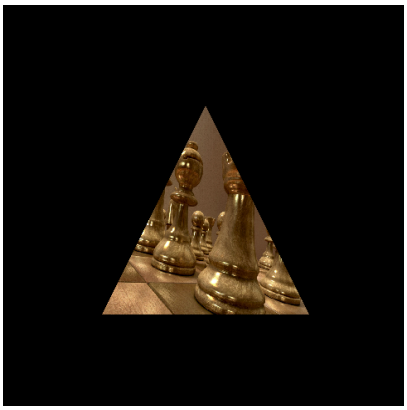
Modifiez la fonction drawCube pour affecter une couleur par face du cube:



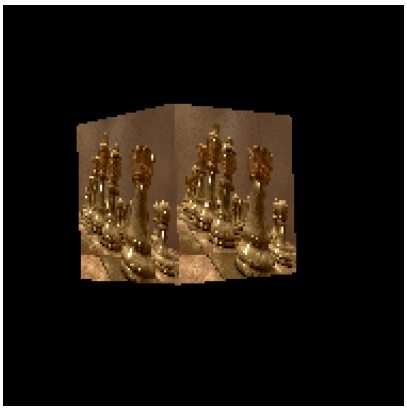
Placage de texture

L'exemple fournit vous permet d'afficher un carré texturé via *./gltest -tx chess.bmp* ou un cube semi-texturé via *./gltest -mode 1 -tx chess.bmp*. Comme vu en cours, openGL permet de spécifier pour chaque vertex d'une primitive les coordonnés u (horizontal), v (vertical) du pixel de la texture à utiliser pour ce vertex. L'interpolation pour le rendu du triangle se fait alors entre les coordonnées de texture de chaque sommet du triangle. Vous utiliserez la primitive glTexCoord2f qui assigne les coordonnées de textre pour les prochains vertex.

Modifiez la fonction display_quad pour déssiner les figures suivantes:



Modifiez la fonction drawCube pour affecter la texture sur une autre face du cube:

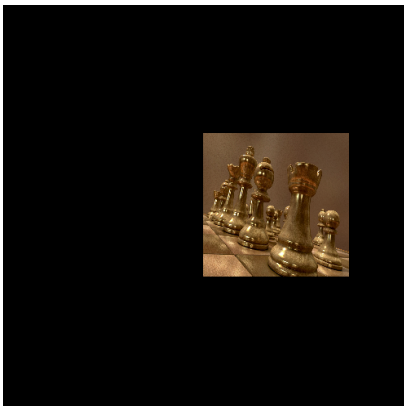


Déplacer des objets

Comme vu en cours, la pluparts des scènes 3D sont composées d'objets que l'on peut positionner relativement les uns aux autres. Dans un premier temps nous allons déplacer un objet seul. Vous utiliserez les fonctions

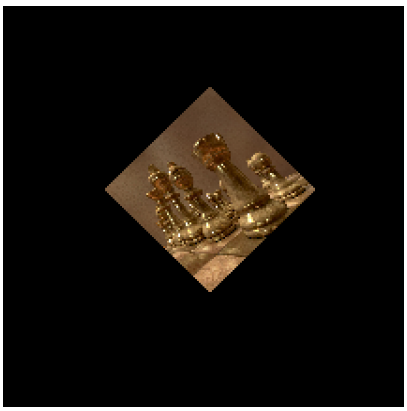
- glRotatef qui permet de tourner les primitives graphiques qui suivent
- glTranslatef qui permet de déplacer les primitives graphiques qui suivent

Modifiez la fonction display3D pour déplacer votre cube:

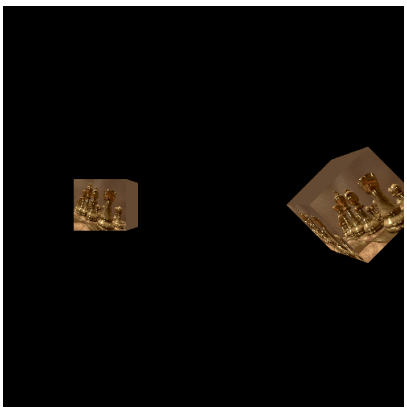


Que se passe-t-il lorsque vous modifiez la translation en z ?

Modifiez la fonction display3D pour tourner votre cube:



Modifiez votre code pour afficher deux cubes:



Vous utiliserez pour cela deux fonctions

- glPushMatrix qui permet de dupliquer la matrice courrante et la mettre comme nouvelle matrice sur la pile des matrices
- glPopMatrix qui permet d'enlever la matrice courante de la pile et restore la martrice précédente

Cette gestion par pile de matrice est typique de OpenGL sans shaders. En

mode shaders vous aurez le plus souvent à gérer vous-mêmes les matrices dans votre programme (coté CPU).

Gestion de la caméra

Les matrices peuvent aussi être utile pour modifier le point de vu global de la scène. Le code que vous avez permet de tourner le modèle selon l'axes des X et des Y.

Modifiez la fonction display3D pour touner le modèle selon l'axe des X et des Z.

Comme vu en cours, deux types de projection existent: projection orthogonale et projection perspective..

Modifiez la fonction display3D pour utiliser une projection orthogonale (on regardera le code de la fonction display2D)

Que remarquez-vous lorsque vous déplacer la caméra ?

Modifiez la fonction display3D pour utiliser une projection perspective en changeant l'ouverture (field of view) via la souris.

Comme vu en cours, OpenGL travaille en coordonnées normalisées [-1,1]* [-1,1]. La conversion en coordonées écrans se fait au dernier moment, en utilisant les instructions données par la fonction glViewport.

Modifiez votre code pour afficher la scene selon deux points de vue décallés uniquement selon l'axe des x.

Félicitations, vous avez un premier code de rendu d'un monde en mode stéréoscopique !

Gestion de la lumière

Modifiez le code display3D pour insérer après la ligne *glDisable(GL_LIGHTING)* un appel à la fonction set_light().
Observez le comportement lorsque vous déplacer la caméra. D'après vous, où se situe la lumière ?
L'exemple de cube donné ne précise pas les normales par face, ce qui explique le résultat. Spécifier pour chaque face la normale via la fonction glNormal3f et regardez le résultat.
Bouger l'appel à la fonction set_light() juste avant la ligne *switch (obj_mode) {*
Qu'observez-vous lors du déplacement de caméra ? Comment l'expliquer?

Shaders OpenGL

Comme vu en cours, les versions moderne d'OpenGL ont la possibilité d'executer des programmes, ou shaders, pour modifier les diverses opérations effectués par le GPU avant projection de la primitive/interpolation en pixel (vertex shader) et au moment de l'écriture du fixel (fragment shader).
Le code fournit permet déjà d'utiliser les shaders:

- cube simple: *./gltest -mode 1 -shader*
- cube texturé: *./gltest -mode 1 -shader -tx chess.bmp*

Nous travaillons tout d'abord en mode cube non texturé

Etudiez la fonction drawCube_VBO et les fichiers glsl_vert.txt et glsl_frag.txt

- à quoi sert la fonction glBufferData ?
-
- à quoi servent les fonctions glGetAttribLocation , glVertexAttribPointer et glEnableVertexAttribArray?
- changer la couleur du cube
- en utilisant la variable openGL gl_FragCoord, proposez un coloriage différent du cube en fonction de la position du pixel dessiné.

Nous travaillons ensuite en mode cube texturé

Etudiez la fonction drawCube_VBO et les fichiers glsl_vert_tx.txt et glsl_frag_tx.txt

- à quoi sert la fonction glBufferData ?
-

- à quoi servent les fonctions glGetAttribLocation , glVertexAttribPointer et glEnableVertexAttribArray?
- comment les pixels de la texture sont-ils récupérés
- changer la couleur du cube
- Proposez un coloriage différent du cube en fonction de la position du pixel dessiné.