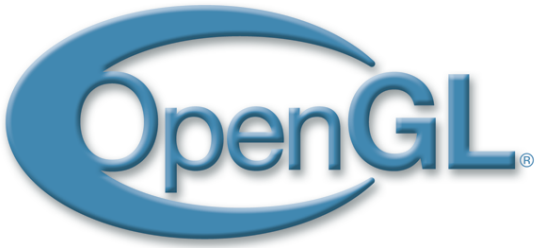
# Introduction à OpenGL avec GLUT

Nikolas Stott

INRIA Saclay - CMAP, École Polytechnique, Université Paris-Saclay

9 février 2017



# Plan

# Qu'est ce qu'OpenGL ?

OpenGL (Open Graphics Library) est une bibliothèque graphique 2D/3D pour des applications 3D (interactives) :

- Interface logicielle bas niveau avec le hardware graphique
- 150 commandes différentes pour spécifier objets et opérations

OpenGL est indépendant du hardware et utilisé dans différents langages à travers différentes bibliothèques :

- GLU/GLUT en C/C++
- Java OpenGL (JOGL) en Java
- WEBGL en Javascript

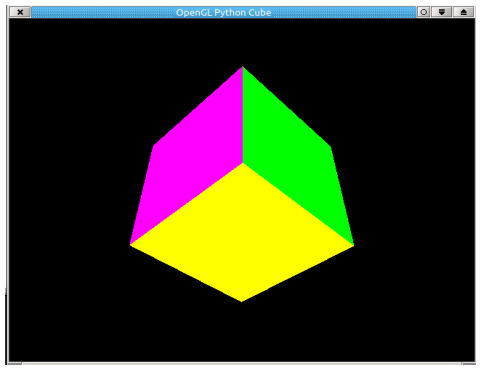
OpenGL ne gère pas le fenêtrage ou l'interface graphique.

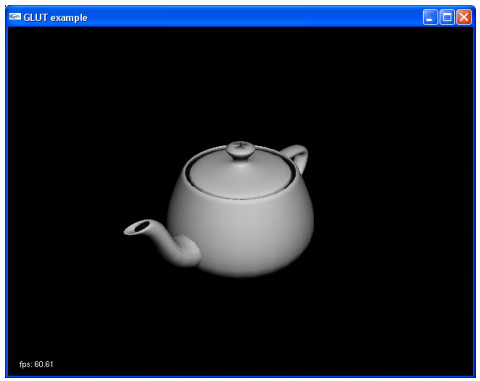
# Qu'est ce que GLUT ?

GLUT (OpenGL Utility Toolkit) est une interface de programmation pour OpenGL qui gère le fenêtrage.
GLUT est simple, petit et utile pour apprendre à découvrir OpenGL.

GLUT contient les fonctionnalités suivantes :

- gestion de l'affichage de fenêtres de rendu OpenGL
- gestion du temps, d'événements et d'interaction utilisateur
- création rapide d'objets primitifs (cube, tétraèdre, sphère, cône, etc) pleins ou maillage seul.

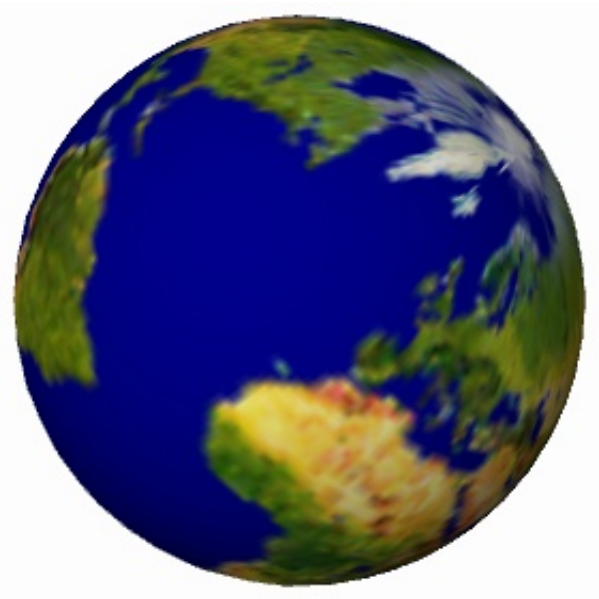


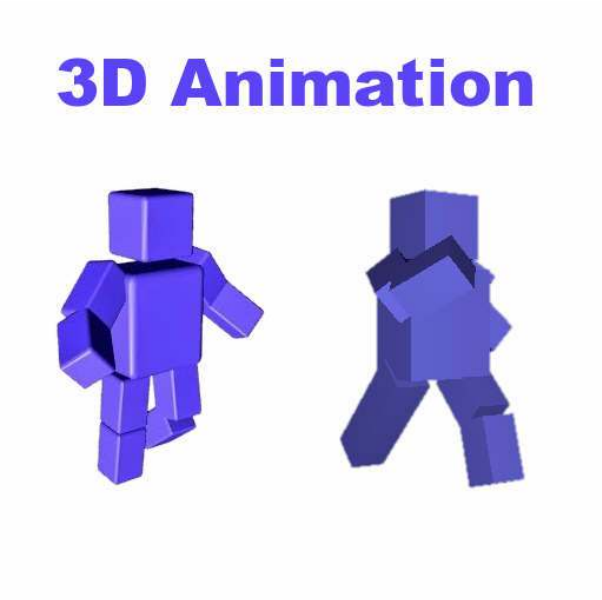



# Qu'est ce qu'OpenGL sait faire ?

## Modélisation/Visualisation

- Création de géométries complexes
- Habillage de la géométrie : couleur, texture, éclairage...
- Visualisation des objets






## Animation

OpenGL permet d'animer :

- la caméra dans la scène
- les objets dans la scène
- le maillage des objets

# La fonction Main

## Contenu

La fonction main doit :

- initialiser GLUT : `glutInit(&argc,argv);`
- paramétrer l'affichage avec `glutInitDisplayMode( ... );` : `GLUT_RGBA`, `GLUT_DEPTH`, `GLUT_SINGLE` ou `GLUT_DOUBLE`
- créer la fenêtre : `glutCreateWindow("C'est bientôt fini ;)");`
- initialiser les variables/objets du programme
- déclarer les fonctions
  - de dessin : `glutDisplayFunc( ... );`
  - de redimensionnement : `glutReshapeFunc( ... );`
  - d'interaction souris : `glutMouseFunc( ... );`
  - d'interaction clavier : `glutKeyboardFunc( ... );`
  - d'évolution autonome : `glutIdleFunc( ... );`
  - de temporisation : `glutTimerFunc( ... );`
- lancer la boucle infinie : `glutMainLoop();`

# Autres fonctions (1)

## Fonction de dessin

Donnée en paramètre de `glutDisplayFunc(...)`
Elle ne prend rien en paramètre.
Son rôle est de tracer l'image courante :

- effacer l'image précédente : `glClear(GL_COLOR_BUFFER_BIT);`
- dessiner ce que l'utilisateur souhaite
- demander de l'afficher : `glFlush();` ou `glSwapBuffers();`

## Fonction de redimensionnement

Donnée en paramètre de `glutReshapeFunc( ... );`
Elle prend en paramètre les dimensions du viewport.
Elle doit assurer la cohérence de la fenêtre de tracé :

- déclarer le viewport : `glViewport(x1,y1,x2,y2);`
- charger les paramètres caméra initiaux

# Autres fonctions (2)

## Fonction d'interaction souris

Donnée en paramètre de `glutMouseFunc( ... );`
Elle prend en paramètre le bouton activé, l'état du bouton et la position écran lors de l'action.

- Le bouton prend les valeurs `GLUT_LEFT/MIDDLE/RIGHT_BUTTON`
- L'état prend les valeurs `GLUT_UP` et `GLUT_DOWN`

## Fonction d'interaction clavier

Donnée en paramètre de `glutKeyboardFunc( ... );`
Elle prend en paramètre la touche activée et la position écran de la souris lors de l'action.

# Autres fonctions (3)

## Fonction d'évolution autonome

Donnée en paramètre de `glutIdleFunc( ... );`
Appelée lorsqu'aucune action n'est déclenchée, elle ne prend aucun paramètre. C'est la fonction qui calcule le nouvel état du système, etc.

## Fonction de temporisation

Donnée en paramètre de `glutTimerFunc( ... );`
Fonction avancée qui permet d'introduire des paramètres temporels dans le programme.
`glutTimerFunc(DeltaT,timer,0)` appelle la fonction de temporisation `timer` au moins toutes les `DeltaT` ms.

## Fonction d'actualisation

`glutPostRedisplay();` est la fonction qui demande à GLUT de calculer et d'afficher une nouvelle image sur l'écran.

# Commandes principales

## Eigen : :Vector3f v

- Vecteur de 3 floats x,y,z : Eigen : :Vector3f(x,y,z)
- Acces aux valeurs : v[i]
- Compatible avec les opérations standards sur des vecteur : 3*u - 2*v est une syntaxe valide

## Produit scalaire $\langle u, v \rangle$

- Calculé par u.dot(v)
- Norme d'un vecteur : sqrt(u.dot(u)) ou u.norm()
- Vecteur normalisé : $\mathrm{u}_{\mathrm{norm}}$ = u.normalized()
- Autre version (inplace) : u.normalize()

## Produit vectoriel $u \wedge v$

- Calculé par u.cross(v)

# TP :

# Simulation d'une flotte de drones