OpenGL, OpenGL/ES, Introduction

Renaud BLANCH
Université Joseph Fourier
Laboratoire d'Informatique de Grenoble

http://iihm.imag.fr/blanch/
mailto:blanch@imag.fr

Plan

0. Introduction
1. Pipeline graphique fixe
2. Pipeline graphique programmable

Plan

0. Introduction
1. Pipeline graphique fixe
2. Pipeline graphique programmable

0. Introduction

0.0 OpenGL ?
0.1 Historique
0.2 OpenGL vue du ciel
0.3 Autour d'OpenGL
0.4 *OpenGL Utility Toolkit*
0.5 Un peu d'algèbre linéaire

0. Introduction

0.0 OpenGL ?
0.1 Historique
0.2 OpenGL vue du ciel
0.3 Autour d'OpenGL
0.4 *OpenGL Utility Toolkit*
0.5 Un peu d'algèbre linéaire

0.0 OpenGL ?

API graphique **indépendante**
• du langage de programmation ;
• du système d'exploitation ;
• du système de fenêtrage,

principalement **dédiée à la 3D**,

permettant de tirer parti **de manière portable**
de l'**accélération matérielle** du rendu graphique.

Il en existe des implémentations purement logicielles (e.g., MESA).

0.0 OpenGL ?

API graphique permettant de manipuler :
• des **primitives** (points, lignes, polygones) ;
• des **images**,

pour **synthétiser des images** dans un *framebuffer* (typiquement l'écran).

7

0.0 OpenGL ?

API graphique **spécifiée** par un consortium,

l'**OpenGL ARB** (*Architecture Review Board*) puis le **Khronos Group**,

composé des différents acteurs impliqués
(fabricants de *chipsets*, de cartes graphiques, d'ordinateurs)

8

0. Introduction

9

0.1 Historique

1992 Silicon Graphics crée
l'OpenGL Architecture Review Board (**ARB**)

1993 OpenGL 1.0 (largement inspirée d'Iris GL)
1997 OpenGL 1.1 *(vertex arrays, texture objects)*
1998 OpenGL 1.2 *(3D textures, imaging, ...)*
2001 OpenGL 1.3 *(cube map, multi-textures, ...)*
2002 OpenGL 1.4 *(automatic mipmap, ...)*
2003 OpenGL 1.5 *(VBO, nonpower-of-2 textures, ...)*

0.1 Historique

2004 **OpenGL 2.0** *vertex & fragment processing*
GLSL 1.10.59 (OpenGL Shading Language)
2006 OpenGL 2.1
GLSL 1.20.8

2008 OpenGL 3.0 *deprecation of the fixed pipeline*
2009 OpenGL 3.1 *removal of deprecated functions*
2009 OpenGL 3.2 *introduction of core (~3.1) and*
GLSL 1.50.09 *compatibility profiles (pre 3.1)*

0.1 Historique

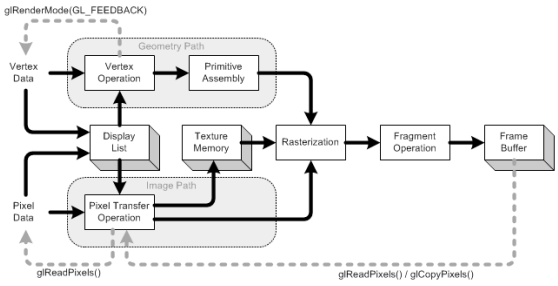
futur convergence :
- GLSL
- OpenCL (*General purpose computing on GPU - GPGPU*)

0. Introduction

0.0 OpenGL ?
0.1 Historique
0.2 OpenGL vue du ciel
0.3 Autour d'OpenGL
0.4 *OpenGL Utility Toolkit*
0.5 Un peu d'algèbre linéaire

0.2 OpenGL vue du ciel

pipeline graphique



http://www.songho.ca/opengl/

0.2 OpenGL vue du ciel

déclarations

```
#include <GL/gl.h>
```

0.2 OpenGL vue du ciel

déclarations

```
#include <GL/gl.h>
```

"namespace"

```
glEnable(GL_BLEND);
const GLfloat v[3] = { 1., 1., 1. };
```


0.2 OpenGL vue du ciel

déclarations

```
#include <GL/gl.h>
```

"namespace"

```
glEnable(GL_BLEND);
const GLfloat v[3] = { 1., 1., 1. };
```

"**polymorphisme**"

```
glVertex3fv(v);
glVertex2i(12, 34);
```


0.2 OpenGL vue du ciel

en **Python** avec **PyOpenGL**

(travaux pratiques)

```
from OpenGL.GL import *

v = (1., 1., 1.)
glVertex(v)
glVertex(12, 34)
```

0. Introduction

0.0 OpenGL ?
0.1 Historique
0.2 OpenGL vue du ciel
0.3 Autour d'OpenGL
0.4 *OpenGL Utility Toolkit*
0.5 Un peu d'algèbre linéaire

0.3 Autour d'OpenGL

OpenGL ES est une version d'OpenGL spécialisée pour les systèmes embarqués.
En pratique c'est un **sous-ensemble** (d'API et de fonctionnalités) d'un version de référence d'**OpenGL**.

OpenGL ES 1.1 (resp. 2.0) est ainsi spécifiée relativement à OpenGL 1.5 (resp. 2.0).

0.3 Autour d'OpenGL

GLU (*OpenGL Utility Library*) est une bibliothèque fournie avec OpenGL qui fournit des fonctionnalités de plus hauts niveaux
(*splines*, quadriques, triangulation, ...)

0.3 Autour d'OpenGL

OpenGL étant indépendant du système hôte,
le **lien avec le système du fenêtrage** passe
par une bibliothèque dépendante du système :
- **GLX** (X Window) ;
- **WGL** (Windows) ;
- **CGL, AGL** (Mac OS X).

Cela peut être caché par les *toolkits* graphiques
portables (Qt, Gtk, **GLUT**, ...)

0. Introduction

0.4 *OpenGL Utility Toolkit*

GLUT fournit le support minimal qui permet
d'expérimenter avec OpenGL :

- **création de fenêtres** et de contextes
  OpenGL associés ;
- **réaction à des événements**
  (gestionnaire de fenêtres, clavier, souris, ...) ;
- **affichage de texte** ;
- utilisation de menus.

0.4 *OpenGL Utility Toolkit*

glut**Init**	initialisation
glut**InitDisplayMode**	
glut**PostRedisplay**	demande un réaffichage
glut**SwapBuffers**	permutation des *buffers*
glut**BitmapCharacter**	affichage de texte
glut**StrokeCharacter**	

0.4 *OpenGL Utility Toolkit*

glut**Reshape**Func	fenêtrage &
glut**Display**Func	réaffichage
glut**Keyboard**Func	clavier ;
glut**Mouse**Func	boutons de la souris ;
glut**Motion**Func	mouvements avec,
glut**PassiveMotion**Func	sans bouton enfoncé
glut**Idle**Func	aussi souvent que possible
glut**Timer**Func	après un délai

0.4 *OpenGL Utility Toolkit*

```
#include <stdlib.h>
#include <GL/glut.h>

void reshape(int width, int height);
void display();
void mouse(int button, int state, int x, int y);

int main(int argc, char *argv[]) {
   glutInit(&argc, argv);
   glutInitDisplayMode(GLUT_RGBA|GLUT_DOUBLE);

   glutCreateWindow(argv[0]);

   glutReshapeFunc(reshape);
   glutDisplayFunc(display);
   glutMouseFunc(mouse);

   scene_init();

   glutMainLoop();

   return EXIT_SUCCESS;
}
```

```
void display() {
   scene_draw();
   glutSwapBuffers();
}

void mouse(int button, int state, int x, int y) {
   if(button == GLUT_LEFT_BUTTON) {
      if(state == GLUT_DOWN) {
         scene_click(x, y);
         glutPostRedisplay()
      }
   }
}
```

0.4 *OpenGL Utility Toolkit*

```
from OpenGL.GLUT import *

def main(argv):
   glutInit(argv)
   glutInitDisplayMode(GLUT_RGBA|GLUT_DOUBLE)

   glutCreateWindow(argv[0])

   glutReshapeFunc(reshape)
   glutDisplayFunc(display)
   glutMouseFunc(mouse)

   scene_init()

   glutMainLoop()
```

```
def display():
   scene_draw()
   glutSwapBuffers()

def mouse(button, state, x, y):
   if button == GLUT_LEFT_BUTTON:
      if state == GLUT_DOWN:
         scene_click(x, y)
         glutPostRedisplay()
```

28

0. Introduction

0.0 OpenGL ?
0.1 Historique
0.2 OpenGL vue du ciel
0.3 Autour d'OpenGL
0.4 *OpenGL Utility Toolkit*
0.5 Un peu d'algèbre linéaire

29

0.5 Un peu d'algèbre linéaire

Les **transformations géométriques** usuelles :

- translation ;
- rotation ;
- changement d'échelle,

peuvent être réalisées grâce à des **opérations matricielles**.

30

0.5 Un peu d'algèbre linéaire

translation

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$

$$= \begin{pmatrix} x+t_x \\ y+t_y \end{pmatrix}$$

translation du point (x, y)
par le vecteur $\vec{t} = (t_x, t_y)$

0.5 Un peu d'algèbre linéaire

mise à l'échelle

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

$$= \begin{pmatrix} x \times s_x \\ y \times s_y \end{pmatrix}$$

0.5 Un peu d'algèbre linéaire

rotation

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

$$= \begin{pmatrix} x\cos\theta - y\sin\theta \\ x\sin\theta + y\cos\theta \end{pmatrix}$$

rotation d'angle θ
autour de l'origine

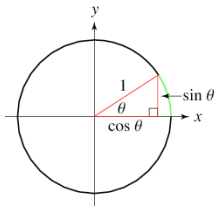
0.5 Un peu d'algèbre linéaire

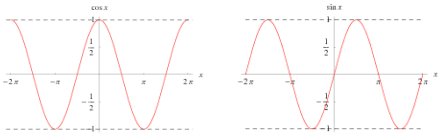
rotation

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \rho\cos\theta \\ \rho\sin\theta \end{pmatrix}$$

$$\begin{pmatrix} \rho \\ \theta \end{pmatrix} = \begin{pmatrix} \sqrt{x^2+y^2} \\ \text{atan2}\,(y,x) \end{pmatrix}$$

coordonnées
cartésiennes vs. **polaires**





http://mathworld.wolfram.com/Trigonometry.html

0.5 Un peu d'algèbre linéaire

En ajoutant une **coordonnée**
dite **homogène**,
on peut travailler uniquement
avec des **produits matriciels**.

0.5 Un peu d'algèbre linéaire

translation

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x+t_x \\ y+t_y \\ z+t_z \\ 1 \end{pmatrix}$$

0.5 Un peu d'algèbre linéaire

mise à l'échelle

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x \times s_x \\ y \times s_y \\ z \times s_z \\ 1 \end{pmatrix}$$

0.5 Un peu d'algèbre linéaire

rotation d'angle θ autour de l'axe $\left|(r_x, r_y, r_z)\right| = 1$

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} r_x^2(1-\cos\theta)+\cos\theta & r_y r_x(1-\cos\theta)-r_z\sin\theta & r_z r_x(1-\cos\theta)+r_y\sin\theta & 0 \\ r_x r_y(1-\cos\theta)+r_z\sin\theta & r_y^2(1-\cos\theta)+\cos\theta & r_z r_y(1-\cos\theta)-r_x\sin\theta & 0 \\ r_x r_z(1-\cos\theta)-r_y\sin\theta & r_y r_z(1-\cos\theta)+r_x\sin\theta & r_z^2(1-\cos\theta)+\cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

qui se simplifie dans certains cas
(e.g., rotation autour de l'axe z)

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

0.5 Un peu d'algèbre linéaire

produit **scalaire** *(dot product)* $\quad \vec{u} = \begin{pmatrix} x_u \\ y_u \\ z_u \end{pmatrix}, \quad \vec{v} = \begin{pmatrix} x_v \\ y_v \\ z_v \end{pmatrix}$

$$\vec{u} \cdot \vec{v} = x_u x_v + y_u y_v + z_u z_v \qquad |\vec{v}| = \sqrt{\vec{v} \cdot \vec{v}}$$
$$= \langle \vec{u}, \vec{v} \rangle$$
$$= |\vec{u}|\,|\vec{v}|\,\cos\left(\widehat{\vec{u}\,\vec{v}}\right)$$
$$= |\vec{u}|\,|\vec{v}| \iff \vec{u} \parallel \vec{v}$$
$$= 0 \iff \vec{u} \perp \vec{v}$$

mesure la **colinéarité**

0.5 Un peu d'algèbre linéaire

produit **vectoriel** *(cross product)*

$$\vec{u} \wedge \vec{v} = \begin{pmatrix} y_u z_v - z_u y_v \\ z_u x_v - x_u z_v \\ x_u y_v - y_u x_v \end{pmatrix}$$
$$= \vec{u} \times \vec{v}$$

$$(\vec{u} \wedge \vec{v}) \cdot \vec{u} = (\vec{u} \wedge \vec{v}) \cdot \vec{v} = 0$$

construit
la **normale** à un plan

$$|\vec{u} \wedge \vec{v}| = |\vec{u}|\,|\vec{v}|\,\sin\left(\widehat{\vec{u}\,\vec{v}}\right)$$
$$= |\vec{u}|\,|\vec{v}| \iff \vec{u} \perp \vec{v}$$
$$= 0 \iff \vec{u} \parallel \vec{v}$$

mesure l'**orthogonalité**