Laboratori IDI
OpenGL

Professors d'IDI, Q2-2012/13

1 Introducció

Aquesta sessió està dedicada a OpenGL, que és una API per al desenvolupament d'aplicacions gràfiques 3D. Començarem amb una introducció a OpenGL que recull aspectes generals i l'esquema bàsic de funcionament. Aquesta introducció sols pretén proporcionar-vos els elements més bàsics que us caldran al llarg de les properes sessions de laboratori. OpenGL és una API complexa, i preferim anar introduint els diferents aspectes a mida que siguin necessaris.

2 Introducció a OpenGL

2.1 Què és OpenGL

OpenGL és una API (*Application Program Interface*) pel desenvolupament d'aplicacions gràfiques 3D. Existeixen implementacions d'OpenGL per gairebé totes les plataformes (Linux, Windows, MacOs ...), fet que fa que OpenGL sigui actualment l'estàndard més utilitzat en la programació d'aplicacions 3D.

La interfície d'OpenGL consisteix en una biblioteca de més de 300 funcions C/C++ (més centenars d'extensions). Encara que l'API està especificada en llenguatge C, existeixen components diversos que permeten cridar funcions d'OpenGL des d'altres llenguatges com ara Python, Perl, Java i més recentment des de JavaScript (WebGL).

Per tal de permetre la visualització d'una escena 3D, OpenGL proporciona bàsicament dos tipus de funcions: funcions per dibuixar un conjunt de *primitives gràfiques*, com ara punts, línies i polígons, i funcions per configurar la manera en què aquestes primitives gràfiques es dibuixen a la pantalla o per consultar la manera en què es faria.

Un aspecte important d'OpenGL (i que és la clau de la seva portabilitat) és que no incorpora cap funció depenent del sistema de finestres ni del sistema operatiu subjacent. Això vol dir que tasques com la gestió de l'entrada de l'usuari, la gestió d'events i la creació i configuració de les finestres és un aspecte extern a OpenGL. En el nostre cas farem servir OpenGL juntament amb GLUT (*OpenGL Utility Toolkit)*[1], però com veurem de seguida aquesta no és pas l'única opció.

2.2 Llibreries relacionades

Les funcions d'OpenGL estan organitzades en dues APIs. La primera proporciona les funcions bàsiques, que comencen amb el prefix *gl*. La segona proporciona utilitats diverses basades en funcions del nucli bàsic; aquesta segona es diu GLU (*OpenGL Utility Library*) i les seves funcions comencen amb *glu*. Podem veure-les com funcions de conveniència que ens faciliten tasques freqüents, però en cap cas són imprescindibles, ja que elles mateixes el que fan és cridar a funcions de la primera llibreria per a aconseguir els seus propòsits.

Donat que OpenGL no incorpora funcions per a la gestió de finestres, caldrà recórrer a d'altres llibreries. En aquest sentit podem utilitzar directament els serveis del sistema de finestres (X-lib, GDU...), o bé utilitzar una eina de més alt nivell per a la construcció d'interfícies d'usuari.

Pel primer cas, existeixen llibreries que permeten connectar OpenGL amb un sistema de finestres determinat, i que només cal fer servir quan no disposem d'altres eines de més alt nivell. En el cas de X-Windows l'API de connexió és la GLX (*OpenGL Extension to the X Window System*), i en sistemes MsWindows l'API és la WGL (*OpenGL Extension to the Microsoft Windows*). En MacOS el *framework* Cocoa proporciona serveis comparables per a integrar OpenGL.

Altrament, si fem servir una eina de construcció d'interfícies gràfiques d'usuari d'alt nivell tenim al nostre abast un elevat nombre d'opcions. D'aquestes, només en citarem dues, que són multiplataforma: GLUT (*OpenGL Utility*

[1] Aquesta llibreria us l'introduïm en un document separat

Toolkit), que proporciona funcions per definir interfícies molt bàsiques, i és la que farem servir en el laboratori, i Qt, que ja coneixeu, que proporciona un *widget* específic per a incrustar finestres d'OpenGL en una interfície.

A la següent taula teniu un resum de les referències a aquestes llibreries, i indicació de les versions més recents a data d'avui, i les URLs on podeu trobar més informació. Pareu atenció al fet que algunes d'aquestes versions es mouen ràpidament, i les que tindreu disponibles al laboratori no seran les darreres versions que es citen. De fet, la programació d'OpenGL en les versions més recents sols és possible en *hardware* molt nou, i a més requereix un coneixement molt més complet abans de començar a programar-hi, pel que ens cenyirem a una versió més antiga però que ens permet una presentació més gradual. Alguns dels aspectes d'OpenGL introduïts en aquest laboratori, però, canvien de forma substancial en progressar a versions més actuals. Més endavant podeu fer l'assignatura *Gràfics* del grau, en què aprendreu el necessari per a fer servir les versions més avançades.

0 Aquest darrer aclariment he dubtat si posar-ho o no... Però no sembla just mentir-lis sense paliatius. Què en penseu?

OpenGL	
Descripció:	API per a crear aplicacions 3D
Ver. actual:	4.3
Web oficial:	`http://www.opengl.org`
GLU	**OpenGL Utility Library**
Descripció:	Utilitats diverses (matrius per càmeres, suport a polígons amb forats...)
Ver. actual:	1.3
Web oficial:	`http://www.opengl.org`
GLX API	**OpenGL Extension to the X Window System**
Ver. actual:	1.4
Descripció:	API per a usar OpenGL amb X-Windows
WGL API	**OpenGL Extension to the Microsoft Windows**
Descripció:	API per a usar OpenGL amb Ms-Windows
GLUT	**OpenGL Utility Toolkit**
Descripció:	API per gestió bàsica de finestres i events.
Ver. actual:	3 (API), 3.7 (codi)
Web oficial:	`http://reality.sgi.com/opengl/glut3/glut3.html`
QT	**Qt system**
Descripció:	API per creació de GUI avançats.
Ver. actual:	4.8
Web oficial:	`http://qt.nokia.com`

2.3 Funcionament bàsic d'OpenGL

Alguns aspectes a tenir presents sobre la construcció d'aplicacions amb OpenGL:

- *Repintat complet de cada frame.* En OpenGL, cada nova imatge es re-dibuixa esborrant el contingut de la finestra i tornant a dibuixar totes les primitives gràfiques de l'escena. Per tant, en una animació amb OpenGL tots els fotogrames es regeneren començant amb la finestra buida, sense aprofitar res del fotograma anterior. Aquest fet és habitual en les aplicacions 3D perquè una variació (fins i tot petita) del punt de vista fa que canviï pràcticament tot el contingut de la vista. Per tal de facilitar l'animació es fa servir la tècnica de doble-*buffering*.

- *Doble-buffering.* OpenGL suporta aquesta tècnica que utilitza dos *buffers* per emmagatzemar la imatge. Un *buffer* és llegit pel hardware gràfic que converteix la imatge en un senyal de vídeo (el *front buffer*) mentre l'altre *buffer* s'utilitza per fer el repintat de l'escena que es mostrarà al següent fotograma (*back buffer*). Habitualment l'API que connecta OpenGL amb el sistema de finestres proporciona un funció *swapBuffers()* que permet intercanviar el front i el back *buffer* cada cop que es completa el repintat d'un fotograma.

- *Màquina d'estats.* OpenGL es comporta com una màquina d'estats: l'estat s'emmagatzema en un conjunt de variables que defineixen el comportament d'altres comandes OpenGL. Això permet fer servir comandes

amb un nombre reduït de paràmetres. El valor d'aquestes variables es pot canviar i consultar mitjançant instruccions d'OpenGL. El color, el gruix de línia i les propietats del material són exemples de variables d'estat, cadascuna de les quals té un valor per defecte.

A grans trets, aquest darrer aspecte ens permet classificar les funcions d'OpenGL en tres grups. Un primer grup estaria format per les **funcions de dibuix** (que modifiquen de forma directa el *framebuffer*). Dins aquest grup estan incloses les funcions per l'esborrat de la finestra (*glclear*) i les funcions pel dibuix de primitives (*glBegin/glEnd*). Un segon grup estaria format per totes les **funcions que modifiquen el valor d'una variable d'estat** (*context gràfic*, en terminologia OpenGL): per exemple, les funcions del tipus *glColor*()* ens permeten modificar la variable d'estat que controla el color amb que es dibuixaran les primitives. Finalment, un tercer grup estaria format per les **funcions que permeten consultar el valor d'una variable d'estat**: *glGet*()*, *glIsEnabled()*, ...

2.4 Sintaxi de les comandes d'OpenGL

Una mateixa funció pot adoptar diferents prototipus que varien únicament en aquests aspectes:

- Nombre i tipus dels paràmetres (int, float, double...)
- Direccionament (referència mitjançant apuntador o valor)

Per exemple, per enviar un vèrtex a OpenGL podem fer servir diferents versions de la funció *glVertex*()*:

```
glVertex[234][bsifd...][v](paràmetres)
```

El dígit indica el nombre de paràmetres (2, 3, 4) i la lletra el tipus de dades. Algunes funcions accepten el sufix *v* per indicar que el paràmetre és un apuntador a un vector. El següent fragment mostra diferents opcions vàlides per cridar la funció *glVertex*()*:

```
float x,y;
short v[3];

// Assignem valors a x, y i v

glVertex2f(x, y);
glVertex3sv(v);
```

La següent taula resumeix els diferents tipus de dades que defineix OpenGL:

Sufix	**Tipus**	**Equivalent típic en llenguatge C**	**Tipus OpenGL**
b	8-bit integer	signed char	GLbyte
s	16-bit integer	short	GLshort
i	32-bit integer	long	GLint, GLsizei
f	32-bit floating-point	float	GLfloat, GLclampf
d	64-bit floating-point	double	GLdouble, GLclampd
ub	8-bit unsigned integer	unsigned char	GLubyte, GLboolean
us	16-bit unsigned integer	unsigned short	GLushort
ui	32-bit unsigned integer	unsigned long	GLuint, GLenum, GLbitfield

L'ús dels tipus OpenGL enumerats a la darrera columna es recomana, ja que afavoreix la portabilitat del codi (en cada plataforma, aquests noms estaran definits com el tipus que correspongui en aquella plataforma).

3 Primitives gràfiques

OpenGL proporciona diverses primitives gràfiques. Aquestes primitives es dibuixen amb un bloc *glBegin-glEnd*, dins el qual es proporcionen els diferents vèrtexs que defineixen la geometria. Aquí teniu un esquema de dibuix d'una primitiva:

```
glBegin(tipus de primitiva);
glVertex3f(x1, y1, z1);
glVertex3f(x2, y2, z2);
```

```
4     glVertex3f(x3, y3, z3);
5     ...
6     glEnd();
```

La funció *glBegin()* permet triar entre diferents tipus de primitives:

Tipus de primitiva	**Interpretació dels vèrtexs**
GL_POINTS	Cada vèrtex es dibuixarà com un punt.
GL_LINES	Cada dos vèrtexs formen un segment.
GL_LINE_STRIP	Els vèrtexs formen una poligonal oberta.
GL_LINE_LOOP	Els vèrtexs formen una poligonal tancada.
GL_TRIANGLES	Cada tres vèrtexs defineixen un triangle.
GL_QUADS	Cada quatre vèrtexs defineixen un quadrilàter.
GL_POLYGON	Els vèrtexs es dibuixen com un polígon.

La següent figura mostra l'aspecte de les primitives que proporciona *glBegin()*:

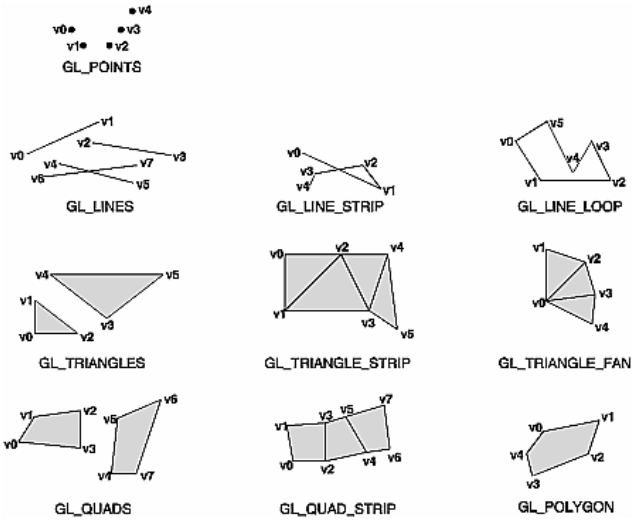

Figura 1: Tipus de primitives.

Tingueu en compta que sols veureu dibuixat a pantalla el que estigui comprés dins del cub format pels punts totes les coordenades dels quals estan entre 0 i 1. Per exemple, el vèrtex a $(0.2, 0.3, -1.0)$ es veurà, però el vèrtex a $(2.0, 0.0, 0.0)$ no.

Hi ha diferents funcions que afecten directament a com es dibuixen els punts (p.ex. *glPointSize*), els segments de línia (*glLineWidth*, *glLineStipple*, *glEdgeFlag*...) i els polígons (*glPolygonMode*, *glPolygonStipple*...). Podeu ampliar la informació sobre aquestes funcions consultant les *man pages* o a la referència online d'OpenGL: `http://www.opengl.org/documentation/blue_book/` També teniu una versió antiga, però suficient pel que us demanarem, a `http://www.lsi.upc.edu/~alvar/opengl`

En les properes sessions de laboratori amb OpenGL us proposarem una sèrie d'experiments que heu de realitzar fent servir aquestes (i algunes altres) funcions, i que us permetran entendre completament el funcionament d'aquests aspectes d'OpenGL.