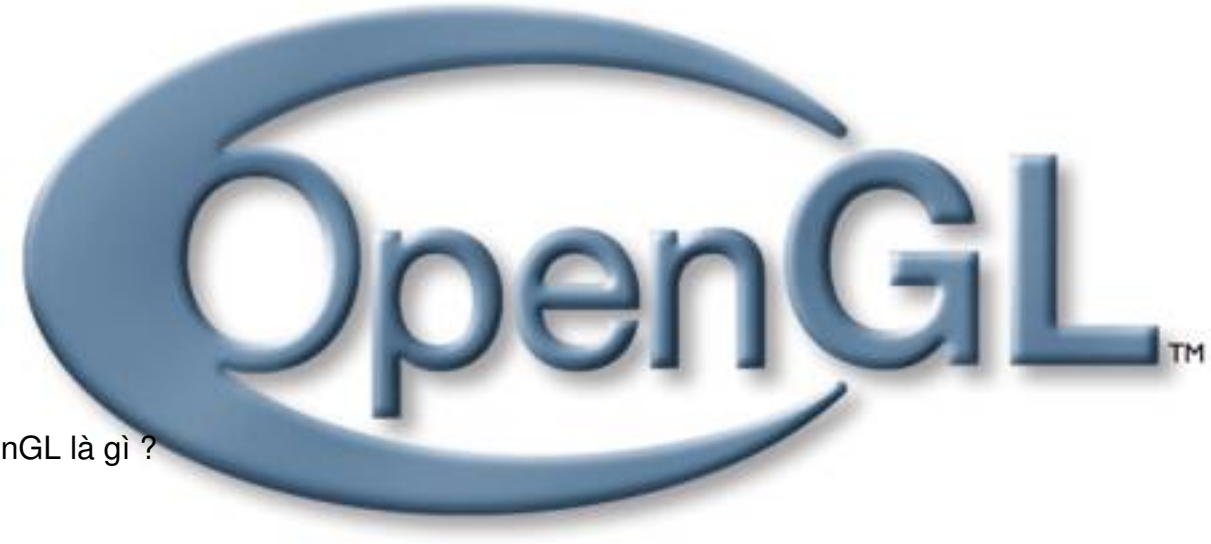

OpenGL là gì ?

Đ□□c phát tri□n đ□u tiên b□i Silicon Graphic, Inc., là m□t giao di□n ph□n ph□n m□m h□□ng th□ t□c theo chu□n công nghi□p h□ tr□ đ□ h□a 3 chi□u. Cung c□p kho□ng 120 tác v□ đ□ v□ các primitive trong nhi□u mode khác nhau. V□i OpenGL, b□n có th□ t□o ra □nh 3 chi□u c□ tĩnh và đ□ng v□i ch□t l□□ng cao.

Là m□t giao di□n ph□n m□m đ□c l□p v□i ph□n c□ng (hardware – independent software interface) h□ tr□ cho l□p trình đ□ h□a. Đ□ làm đ□□c đi□u này, OpenGL không th□c hi□n các tác v□ thu□c v□ h□ đi□u hành cũng nh□ không nh□n d□ li□u nh□p c□a ng□□i dùng (ng□□i dùng giao ti□p v□i OpenGL thông qua OpenGL API). Nó là l□p trung gian gi□a ng□□i dùng và ph□n c□ng. Nghĩa là nó giao ti□p tr□c ti□p v□i driver c□a thi□t b□ đ□ h□a.

GLUT (pronounced like the glut in gluttony) is the OpenGL Utility Toolkit, a window system independent toolkit for writing OpenGL programs. It implements a simple windowing application programming interface (API) for OpenGL. GLUT makes it considerably easier to learn about and explore OpenGL programming. GLUT provides a portable API so you can write a single OpenGL program that works across all PC and workstation OS platforms.

Download phiên b□n m□i nh□t c□a glut □ đây.
Gi□i nén và đ□t các file vào đúng v□ trí:

- glut32.dll vào C:WINDOWSsystem
  - glut.lib vào C:Program FilesMicrosoft Visual Studio 9.0 VC lib
  - glut.h vào C:Program FilesMicrosoft Visual Studio 9.0 VC Include GL

T□t nhiên GLUT là th□ vi□n ph□ thu□c OpenGL cho nên c□n có gl.h, glu.h,glu32.dll, opengl32.dll, opengl32.lib, glu32.lib n□a. T□i t□i đây . Gi□i nén và đ□t vào các v□ trí gi□ng nh□ trên.

Add dòng sau đây vào tr□□c hàm main()
  #pragma comment(linker, "/subsystem:"windows" /entry:"mainCRTStartup"")

Xong, bây gi□ tui s□ h□□ng d□n b□n v□ m□t hình tam giác v□i GLUT

```
#include <GL/glut.h>
#pragma comment( linker, “/subsystem:”windows” /entry:”mainCRTStartup”" )
void renderScene(void) {
glClear(GL_COLOR_BUFFER_BIT);
glBegin(GL_TRIANGLES);
glVertex3f(-0.5,-0.5,0.0);
glVertex3f(0.5,0.0,0.0);
glVertex3f(0.0,0.5,0.0);
glEnd();
glFlush();
}  void main(int argc, char **argv) {
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_DEPTH | GLUT_SINGLE | GLUT_RGBA);
glutInitWindowPosition(100,100);
glutInitWindowSize(320,320);
glutCreateWindow(“Ve hinh tam giac!!!!”);
glutDisplayFunc(renderScene);
glutMainLoop();
}
```

M□t s□ quy □□c (convention) v□ tên hàm OpenGL:

Tên hàm th□ vi□n OpenGL có hình th□c nh□ sau:

Gl{tên hàm}[{s tham s }{lo i tham s }]

Ví d : glClearColor(), glColor3f()

Ph n tên hàm đ c vi t hoa cho nh ng ch cái đ ng đ u 1 t trong tên hàm và ít nhi u nói lean đ c ch c năng c a hàm. Ph n s tham s và lo i tham s xu t hi n tùy theo hàm. S tham s cho ta bi t s l ng tham s s đ a vào khi g i hàm.

Gi i thích các hàm:

void glutInit(int *argc, char **argv); –> //Kh i đ ng GLUT , argc, argv là 2 đ i s dòng l nh c a hàm main

void glutInitWindowPosition(int x, int y); —> //Kh i t o v trí b t đ u c a s , x là left of the screen, y là top of the screen, nói chung đây là đi m bên trái, phía trên c a c a s , t đây ta kéo xu ng phía d i, bên ph i là đc 1 c a s . Đ n v c a x, y là pixel.

void glutInitWindowSize(int width, int height); --> //Kh i t o kích th c c a s . v i chi u dài và chi u r ng, c ng thêm 1 đi m b t đ u m i nói trên n a, b n đã t ng t ng ra đc 1 cái c a s ch a

void glutInitDisplayMode(unsigned int mode) –> //đ nh nghĩa mode hi n th , ch ra màu c a mode và s + ki u c a buffer
+ GLUT_RGBA or GLUT_RGB : c a s màu RGBA, đây là mode m c đ nh
+ GLUT_SINGLE : c a s buffer đ n
+ GLUT_DOUBLE : c a s buffer đôi
+ GLUT_DEPTH : c a s buffer sâu

int glutCreateWindow(char *title); –> t□o c□a s□ có tiêu đ□ title

Bây gi□ ta c□n t□o m□t hàm đ□ trình di□n. Hàm này là do ng□□i dùng t□o ra.

```
void renderScene(void) {
glClear(GL_COLOR_BUFFER_BIT); //xóa màn hình
glBegin(GL_TRIANGLES); //v□ tam giác
glVertex3f(-0.5,-0.5,0.0);
glVertex3f(0.5,0.0,0.0);
glVertex3f(0.0,0.5,0.0);
glEnd(); //k□t thúc
glFlush();
}
```

void glutDisplayFunc(void (*func)(void)); –> Hàm này báo cho GLUT bi□t ph□i trình di□n theo hàm nào , đ□i s□ c□a nó là m□t con tr□ hàm tr□ v□ ki□u void

void glutMainLoop(void) –> cu□i cùng ta ph□i l□p l□i hàm main liên t□c đ□ "trình di□n hình tam giác". Gi□ng nh□ ng□□i ta làm phim ho□t hình đó, các frame n□i ti□p nhau trên m□t màn □nh.

Hic, m□y cái này d□ch t□ english sang, ko hi□u l□m, c□ t□m bi□t v□y đã.

****

THAM KH□O THÊM

****

Các l□nh glClearColor(), glClear(),glFush() là nh□ng l□nh c□ b□n c□a Opengl.glClearColor() có nhi□m v□ ch□n màu đ□ xoá window, b□n d□ dàng nh□n ra là nó có 4 tham s□, 4 tham s□ đó là RGBA(red green blue alpha).Không gi□ng v□i hàm RGB() trong Win32 API , 4 tham s□ này có giá tr□ trong kho□ng 0.0f đ□n 1.0f(ki□u float).

Các giá tr□ R,G,B trong OpenGL thì >=0.0 (không có) và <=1.0 (đ□ sáng c□c đ□i). Ba tham s□ đ□u là màu đ□ xanh lá cây và xanh da tr□i, còn tham s□ th□ 4 là đ□ sáng t□i c□a window.Bây gi□ hãy thay đ□i các giá tr□ c□a màu xem th□!Hàm glClear() m□i th□c s□ xoá window, nó có

nh□ng h□ng s□ xác đ□nh.Có tr□□ng h□p có nh□ng hàm ch□a đ□□c ch□y đ□n khi k□t thúc ch□□ng trình, đ□ tránh tr□□ng h□p này hàm glFlush()đ□□c g□i, nó s□ th□c hi□n t□t c□ các hàm ch□a đ□□c ch□y và k□t thúc ch□□ng trình.

Dùng glClear*() đ□ đ□nh màu xóa cho các buffer.
Sau đó glClear(GLbitfied mask) đ□ xóa buffer t□□ng □ng v□i mask
Tham s□ param c□a hàm glClear() có th□ nh□n t□ 1 đ□n 4 giá tr□ sau:

Buffer	**Name**
Color buffer	**GL_COLOR_BUFFER_BIT**
Depth buffer	**GL_DEPTH_BUFFER_BIT**
Accumulation buffer	**GL_ACCUM_BUFFER_BIT**
Stencil buffer	**GL_STENCIL_BUFFER_BIT**

M□i giá tr□ trên ch□ đ□n m□t buffer t□□ng □ng nh□ trên b□ng. Các giá tr□ có th□ k□t h□p thông qua toán t□ '|' (bitwise-OR). Khi mu□n xóa nhi□u b□ đ□m đ□ng th□i, nên dùng bitwise-OR h□n là g□i tách riêng t□ng l□nh glClear() m□t tham s□ vì hàm glClear() có th□ th□c s□ xóa đ□ng th□i nhi□u buffer (ch□c năng này tùy thu□c ph□n c□ng).
Đ□ đ□nh màu xóa cho m□i buffer, ta dùng các hàm glClear*() nh□ sau: glClearColor(), glClearDepth(), glClearAccum(), glClearStencil().

Bu□c vi□c v□ hoàn t□t

Đ□i v□i các □ng d□ng đ□ h□a ch□y qua m□ng, trong đó client ch□y ph□n ch□□ng trình chính và hi□n th□ k□t qu□ đ□n server, th□□ng thì client s□ gom nhi□u l□nh v□ vào m□t packet, sau đó m□i g□i đ□n server. Nh□ng làm sao đ□ client bi□t đ□□c khi nào thì □nh trên server đã v□ xong và g□i ti□p packet khác? Do đó nó s□ đ□i đ□n khi nào packet đ□y m□i g□i ti□p. Nh□ng packet có th□ không bao gi□ đ□y vì vi□c v□ bên client đã hoàn t□t và nh□ v□y server không th□y đ□□c tr□n vein k□t qu□ v□ OpenGL cung c□p hàm glFlush() đ□ chúng ta gi□i quy□t v□n đ□ này. L□nh này bu□c client g□i packet ngay c□ khi packet ch□a đ□y. L□nh này không đ□i cho vi□c v□ hoàn t□t, nó bu□c vi□c v□ ph□i b□t đ□u th□c hi□n và do đó đ□m b□o t□t c□ các l□nh tr□□c đó th□c hi□n trong m□t th□i gian gi□i h□n. N□u máy ch□y local thì ta không c□n dùng l□nh này.

M□t l□nh khác cũng g□n gi□ng là l□nh glFinish(), nó th□c hi□n ch□c năng gi□ng glFlush() nh□ng

nó đ□i ph□n h□i t□ ph□n c□ng đang v□ ho□c m□ng đ□ kh□ng đ□nh vi□c v□ đã hoàn t□t. L□nh này h□u d□ng khi ta mu□n đ□ng b□ hóa các ph□n v□. Ví d□ ta mu□n m□t ph□n nào đó v□ hoàn t□t xong thì m□i th□c hi□n các ph□n khác. Nh□ng l□m d□ng l□nh này s□ làm ch□m ch□□ng trình vì nó ph□i đ□i ph□n h□i.

C□u trúc vertex

Trong OpenGL, m□i đ□i t□□ng hình h□c đ□u đ□□c mô t□ c□ b□n t□ các vertex. C□u trúc này bao g□m b□ 4 s□ th□c ch□ t□a đ□ trong không gian. Đ□ ch□ đ□nh m□t vertex, ta dùng:

glVertex{234}{sifd}[v](TYPEcoords);

Vì sao có s□ 4? Vì OpenGL dùng t□a đ□ t□□ng đ□i theo m□t s□ th□ t□. Khi ta ghi t□a đ□ d□ng (x,y,z,w) thì s□ th□c là t□a đ□ 3 chi□u (x/w,y/w,z/w). M□c đ□nh w=1.0, z=0.

Ví d□:

```
glVertex2s(2, 3);                       // khai báo vertex có t□a đ□ (2,3,0)
glVertex3d(0.0, 0.0, 3.1415926535898);  // to□ đ□ nh□ tham s□
glVertex4f(2.3, 1.0, -2.2, 2.0);        // t□a đ□ (1.15,0.5,1.1)
GLdouble dvect[3] = {5.0, 9.0, 1992.0}; //khai báo 1 vecteor
glVertex3dv(dvect);                     //khai báo vertex thông qua vector nêu trên
```

Trong m□t s□ máy tính, vi□c truy□n tham s□ b□ng vector s□ hi□u qu□ h□n truy□n b□ 3 s□ riêng l□ (tùy thu□c ph□n c□ng).

Đ□ t□o m□t đ□i t□□ng hình h□c t□ các vertex, ta bao kh□i khai báo vertex b□ng hai hàm glBegin(param) và glEnd(). Tham s□ param đ□a vào cho hàm glBegin() s□ giúp OpenGL quy□t đ□nh v□ gì t□ các vertex khai báo bên trong.

Ví d□:

```
glBegin(GL_POLYGON);
glVertex2f(0.0, 0.0);
glVertex2f(0.0, 1.0);
glVertex2f(0.5, 1.0);
glVertex2f(1.0, 0.5);
glVertex2f(0.5, 0.0);
glEnd();
```

Các tham s có th đa vào hàm glBegin(Glenum mode)

Giá tr **Ý nghĩa**

GL_POINTS T ng vertex đ c v riêng
GL_LINES M i c p vertex đ c coi nh 2 đ u m t đo n th ng
GL_POLYGON Các vertex đ c xem nh biên c a m t đa giác l i
GL_TRIANGLES B 3 vertex đ c xem các đ nh m t tam giác (không l p)
GL_QUADS B 4 vertex đ c xem nh 4 đ nh 1 đa giác (không l p).
GL_LINE_STRIP M t lo t các đo n đ c n i v i nhau (có l p)
GL_LINE_LOOP Nh trên nh ng vertex đ u và cu i đ c n i v i nhau (l p vòng)
GL_TRIANGLE_STRIP B 3 vertex xem nh các đ nh tam giác (có l p)
GL_TRIANGLE_FAN Vertex 0 là đ nh chung k t h p c p đ nh b t k t o tam giác

S u t m & ch nh s a