

Tutorial 1 : Initialisation d'OpenGL (Windows)

Pour commencer, il faut créer une application win32 vide. Créez ensuite un fichier *.cpp qui va contenir votre code source. Les includes nécessaires à ce programme sont : iostream, windows.h, math.h et glut.h. Les Libs sont : opengl32.lib, glut32.lib et glu32.lib.

```
#include<iostream> // entrées sorties et fonctions cpp
#include<windows.h> // pour gérer les fonctions windows
#include "glut.h" // pour utiliser les fonctions opengl
#include<math.h> // pour accéder aux fonctions math

#define M_PI 3.1415926535897932384626433832795 // défini pi s'il ne l'est pas dans math.h
float width = 640.0f, height = 480.0f; // largeur et longueur de la fenêtre
HDC DC;
HGLRC RC;
```

La fonction main sera :

```
int WINAPI WinMain( HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
{
WNDCLASS WindowClass =
{ CS_CLASSDC, //Style
WinProc, //Procédure pour la gestion des messages
0,
0,
hInstance, //Handle du programme
0, //Icone
0, //Curseur
0, //Couleur d'arrière-plan
NULL, //Pointeur sur le menu associé la classe
"La classe!" //Nom de la classe fenêtre
};

if (!RegisterClass(&WindowClass))
exit(1);

HWND OpenGLWindow = CreateWindow
( "La classe!", //Classe de la fenêtre
"My Window", //Nom de la fenêtre
WS_VISIBLE | WS_BORDER | WS_OVERLAPPEDWINDOW, //Caractéristiques
0, //Position x
0, //Position y
width, //Largeur
height, //Hauteur
0,
0,
hInstance, //HINSTANCE du programme
```

```

NULL
);

if (!OpenGLWindow)
exit(1);

init(OpenGLWindow);
initGL();

MSG msg;
ZeroMemory( &msg, sizeof(msg) );

// boucle du programme
while(true){
if(PeekMessage(&msg,NULL,0,0,PM_REMOVE)){
if(msg.message==WM_QUIT)
break;
TranslateMessage (&msg) ;
DispatchMessage (&msg) ;
}
else{
Draw();
}

return 0;
}

```

Fonction window proc pour la gestion des messages de windows :

```

LRESULT CALLBACK WinProc( HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam )
{
switch(uMsg)
{
case WM_CLOSE:
wglGetCurrent(NULL, NULL);
if (RC)
wglDeleteContext(RC);
ReleaseDC(hwnd,DC); //Libère le DC et ferme le programme
PostQuitMessage(0);
break;
case WM_CREATE:
DC=GetDC(hwnd); //Récupère le DC
break;
case WM_SIZE:
Reshape(LOWORD(lParam),HIWORD(lParam)); //Met à jour les paramètres OGL
break;

default:
return DefWindowProc(hwnd,uMsg,wParam,lParam);
}

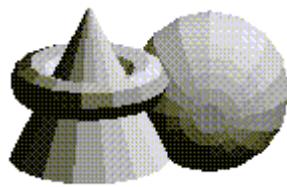
```

```
break;  
}  
return 0;  
}
```

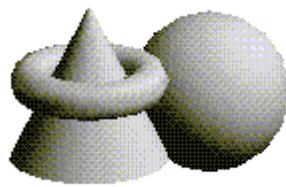
Votre fonction d'initialisation peut ressembler à ça. Si vous voulez des détails sur les paramètres possibles de chaque fonction d'OpenGL, je vous conseille d'aller voir dans le fichier : glut.h.

```
void initGL()  
{  
glPolygonMode(GL_FRONT_AND_BACK,GL_FILL); // mode d'affichage des polygones  
// GL_FRONT_AND_BACK = affiche les 2 cotés  
// GL_FILL = remplit les polygones  
glShadeModel(GL_SMOOTH); // modèle d'illumination GL_FLAT  
  
glEnable(GL_DEPTH_TEST); // activation du Z-Buffer  
glDepthFunc(GL_LEQUAL); // gestion du Z-Buffer : GL_LESS GL_LEQUAL GL_ALWAYS  
// GL_EQUAL GL_GREATER GL_NOTEQUAL GL_GEQUAL  
glEnable(GL_COLOR_MATERIAL); // active la coloration des objets  
  
glClearColor(0.0f,0.0f,0.0f,1.0f); // couleur du fond de la scène  
// init de la matrice de projection  
glViewport(0, 0, width, height);  
glMatrixMode(GL_PROJECTION);  
glLoadIdentity(); // init à la matrice identité  
gluPerspective(45.0f, // angle d'ouverture de la caméra  
width/height, // ratio de la fenêtre  
0.1f,1000.0f // distance min et max de la scène  
);  
// init de la matrice de vue  
glMatrixMode(GL_MODELVIEW);  
glLoadIdentity(); // init à la matrice identité  
gluLookAt(0.0f, 0.0f, 0.0f, // position oeil  
0.0f,0.0f,1.0f, // point regardé  
0.0f,1.0f,0.0f); // vecteur haut  
}
```

Différence entre les modèles d'illumination :



GL_FLAT



GL_SMOOTH

Et enfin la fonction d'affichage :

```

void Draw()
{
// vide le Z-Buffer et le Buffer de couleur
glClear(GL_DEPTH_BUFFER_BIT | GL_COLOR_BUFFER_BIT);

glBegin(GL_TRIANGLES); // mode de polygones à afficher
// couleurs et points des vertexes à afficher
	glColor4f(1.0f,0.0f,0.0f,1.0f); glVertex3f(0.0f, 1.0f, 10.0f);
	glColor4f(0.0f,1.0f,0.0f,1.0f); glVertex3f(-1.0f, -1.0f, 10.0f);
	glColor4f(0.0f,0.0f,1.0f,1.0f); glVertex3f(1.0f, -1.0f, 10.0f);
glEnd();

SwapBuffers(DC); //Echange les 2 frame buffers
}

void Reshape(int w, int h)
{
// re-init de la dimension de la fenêtre
width = w;
height = h;

glViewport(0, 0, width, height);

glMatrixMode(GL_PROJECTION);
glLoadIdentity(); // init à la matrice identité
gluPerspective(45.0f, // angle ouverture caméra
width/height, // ratio de la fenêtre
0.1f,1000.0f // distance min et max de la scène
);
}

```

Plusieurs modes de polygones sont disponibles, en voici la liste :

