OpenGL 3.0

Chapitre 3

Primitives géométriques

C.Turrier
15 décembre 2020

Sommaire

Pour créer des objets 3D, le programmeur utilise des primitives géométriques.
OpenGl offre au programmeur la possibilité d'utiliser les types de primitives géométriques suivants :

- GL_POINTS
- GL_LINES
- GL_LINE_STRIP
- GL_LINE_LOOP
- GL_TRIANGLES
- GL_TRIANGLE_STRIP
- GL_TRIANGLE_FAN
- GL_POLYGON

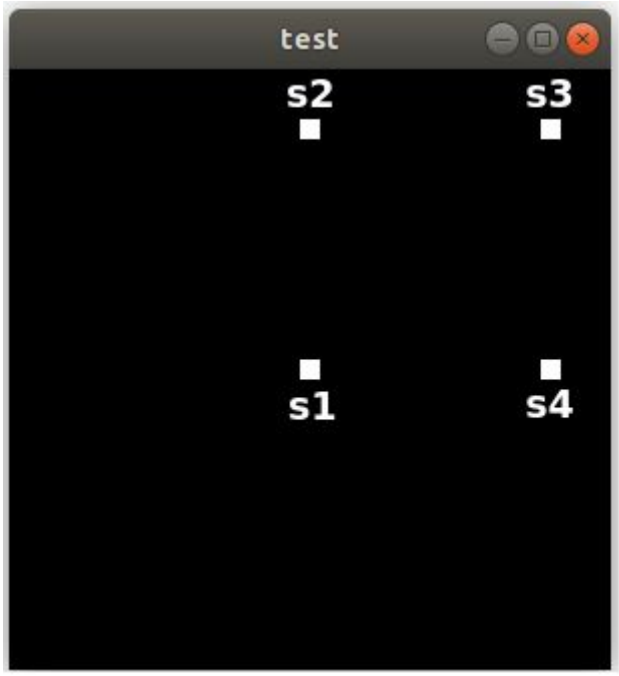
Remarque
Le squelette du programme, suivant, utilisé pour le type GL_POINTS, sera utilisé à l'identique pour tester les autres types de primitives.
Seules les instructions situées entre les instructions glBegin() et glEnd() seront modifiées.

1) Type GL_POINTS

Le type de primitive **GL_POINTS** permet de créer des points.
Chaque sommet est défini par ses coordonnées x,y et z et représente un point.
Les coordonnées x,y et z d'un sommet sont ses coordonnées dans l'espace du modèle ; le modèle étant l'objet 3D définit entre les instructions glBegin() et glEnd().

Exemple
Le programme suivant dessine 4 points.

```
/*
* test GL_POINTS
* compilation:  g++ test.cpp -o test -lglut -lGLU -lGL
* exécution: ./test
*/
#include <GL/freeglut.h>
void affiche(void)
{
glClear(GL_COLOR_BUFFER_BIT);
glColor3f(1.0, 1.0, 1.0);
glPointSize(10.0);
glBegin(GL_POINTS);
glVertex3f(0.0, 0.0, 0.0);//s1
glVertex3f(0.0, 120.0, 0.0);//s2
glVertex3f(120.0, 120.0, 0.0);//s3
glVertex3f(120.0, 0.0, 0.0);//s4
glEnd();
glFlush();
}
void resize(int w, int h)
{
glViewport(0, 0, w, h);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glOrtho(-150.0, 150.0, -150.0, 150.0, -1.0, 1.0);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
}
// Main routine.
int main(int argc, char **argv)
{
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_SINGLE | GLUT_RGBA);
glutInitWindowSize(300, 300);
glutInitWindowPosition(100, 100);
glutCreateWindow("test");
glutDisplayFunc(affiche);
glutReshapeFunc(resize);
glClearColor(0.0, 0.0, 0.0, 0.0);
glutMainLoop();
}
```



Remarques

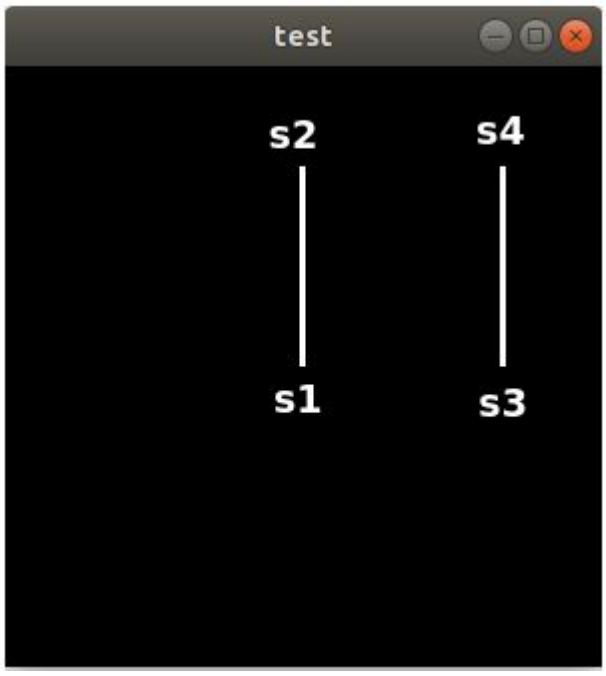
```
glColor3f(1.0, 1.0, 1.0);// couleur des point = blanc
glPointSize(10.0); //taille des points
glBegin(GL_POINTS);
glVertex3f(0.0, 0.0, 0.0); //point bas, gauche, au centre de la fenêtre
glVertex3f(0.0, 120.0, 0.0);//point haut gauche
glVertex3f(120.0, 120.0, 0.0);//point haut droit
glVertex3f(120.0, 0.0, 0.0); //point bas droit
glEnd();
```

2) Type GL_LINES

Le type de primitive **GL_LINES** permet de créer des lignes. Chaque ligne est définie par deux sommets successifs.
Chaque sommet est défini par ses coordonnées x,y et z qui sont ses coordonnées dans l'espace du modèle ; le modèle étant l'objet 3D définit entre les instructions glBegin() et glEnd().
Une succession de sommet s1, s2, s3, s4, s5, s6,... définit les lignes (s1,s2), (s3,s4), (s5,s6), ...

Exemple
Le programme suivant trace deux lignes (le reste du code source est inchangé par rapport à GL_POINTS)

```
glLineWidth(3.0);
glBegin(GL_LINES);
glVertex3f(0.0, 0.0, 0.0);//s1
glVertex3f(0.0, 100.0, 0.0);//s2
glVertex3f(100.0, 0.0, 0.0);//s3
glVertex3f(100.0, 100.0, 0.0);//s4
glEnd();
```

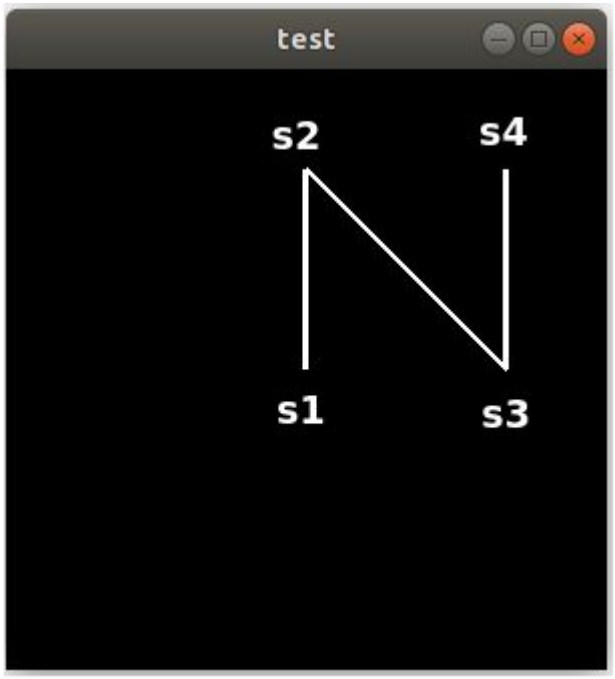


3) Type GL_LINE_STRIP

Le type de primitive **GL_LINE_STRIP** permet de créer une bande de lignes. Une succession de sommet s1, s2, s3, s4, s5, s6,... définit une bande de lignes (s1,s2), (s2,s3), (s3,s4), (s4,s5), (s5, s6),...
Chaque sommet est défini par ses coordonnées x,y et z qui sont ses coordonnées dans l'espace du modèle ; le modèle étant l'objet 3D définit entre les instructions glBegin() et glEnd().

Exemple
Le programme suivant trace une bande de trois lignes, définie par 4 sommets (le reste du code source est inchangé par rapport à GL_POINTS)

```
glLineWidth(3.0);
glBegin(GL_LINE_STRIP);
glVertex3f(0.0, 0.0, 0.0);\\s1
glVertex3f(0.0, 100.0, 0.0);\\s2
glVertex3f(100.0, 0.0, 0.0);\\s3
glVertex3f(100.0, 100.0, 0.0);\\s4
glEnd();
```

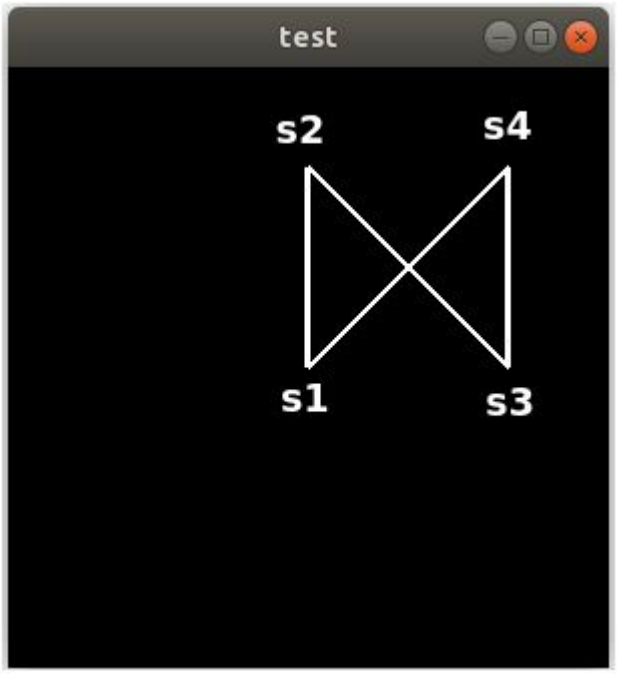


4) Type GL_LINE_LOOP

Le type de primitive **GL_LINE_LOOP** permet de créer une ligne en forme de boucle. Il est identique au type GL_LINE_STRIP avec comme seule différence que le dernier point de la liste est relié par une ligne au premier point de liste.
Une succession de sommet s1, s2, s3, s4, s5, s6,...sn définit une bande de lignes (s1,s2), (s2,s3), (s3,s4), (s4,s5), (s5, s6),...(sn-1,sn), (sn,s1)

Exemple
Le programme suivant crée une ligne en boucle constituée de 4 sommets..

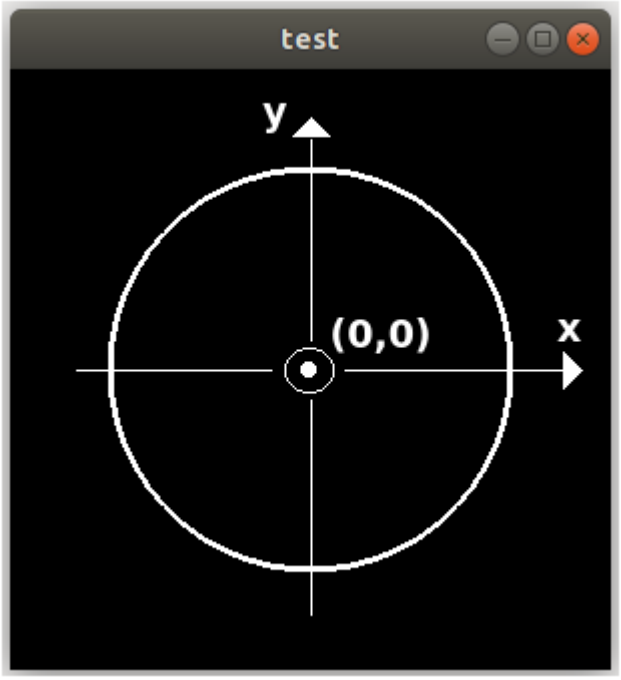
```
glLineWidth(3.0);
glBegin(GL_LINE_LOOP);
glVertex3f(0.0, 0.0, 0.0);
glVertex3f(0.0, 100.0, 0.0);
glVertex3f(100.0, 0.0, 0.0);
glVertex3f(100.0, 100.0, 0.0);
glEnd();
```



Le type de primitive **GL_LINE_LOOP** permet en aprticulier de créer des polygones réguliers à n cotés qui conduisent à des approximations de cercles lorsque le nombre n de cotés est grand.

Exemple
Le programme suivant crée une ligne en boucle constituée de 50 sommets disposés en forme de cercle centré en x,y= (0,0) dans le plan z=0.
Les fonctions void resize(int w, int h) et int main(int argc, char **argv) sont inchangées.

```
/*
* test cercle
* compilation:  g++ test.cpp -o test -lglut -lGLU -lGL
* exécution: ./test
*/
#include <GL/freeglut.h>
#include <math.h>
float R = 100.0; //rayon.
float X = 0.0; float Y = 0.0; // centre
int n = 50; // nombre de sommets
void affiche(void)
{
float theta = 0;
int i;
glClear(GL_COLOR_BUFFER_BIT);
glColor3f(1.0, 1.0, 1.0);
glLineWidth(3.0);
glBegin(GL_LINE_LOOP);
for (i = 0; i < n; ++i)
{
glVertex3f(X+R*cos(theta), Y+R*sin(theta), 0.0);
theta + = 2*M_PI/n;
}
glEnd();
glFlush();
}
```

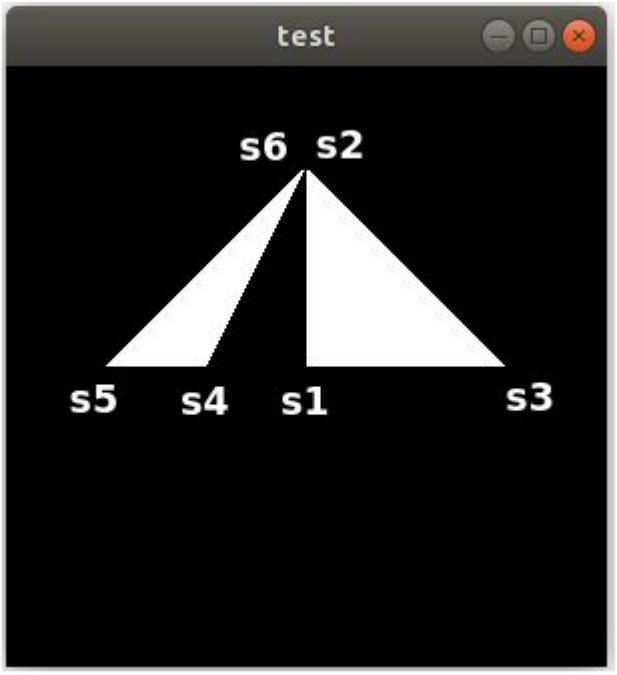


5) Type GL_TRIANGLE

Le type de primitive **GL_TRIANGLE** permet de créer des triangles.
Chaque triangle est défini par trois sommets successifs.
Chaque sommet est défini par ses coordonnées x,y et z qui sont ses coordonnées dans l'espace du modèle ; le modèle étant l'objet 3D définit entre les instructions glBegin() et glEnd().
Une succession de sommet s1, s2, s3, s4, s5, s6 définit deux triangles (s1,s2,s3) et (s4,s5,s6)

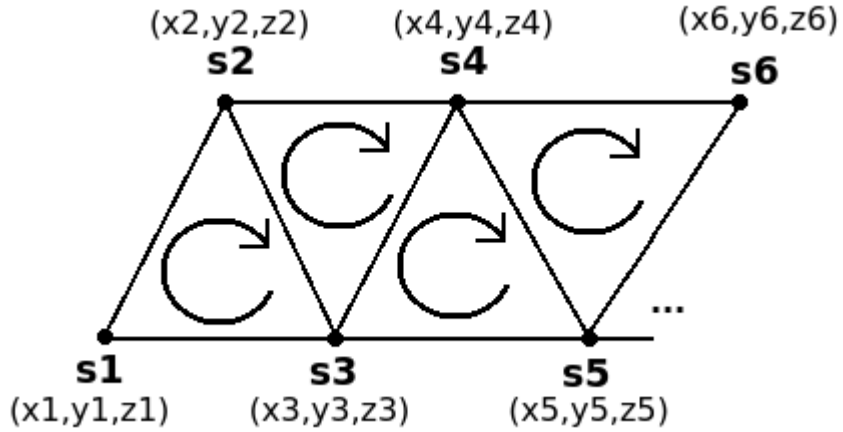
Exemple
Le programme suivant crée deux triangles.

```
glBegin(GL_TRIANGLES);
glVertex3f(0.0, 0.0, 0.0);//s1
glVertex3f(0.0, 100.0, 0.0);//s2
glVertex3f(100.0, 0.0, 0.0);//s3
glVertex3f(-50.0, 0.0, 0.0);//s4
glVertex3f(-100.0, 0.0, 0.0);//s5
glVertex3f(0.0, 100.0, 0.0);//s6
glEnd();
```



6) Type GL_TRIANGLE_STRIP

Le type de primitive GL_TRIANGLE_STRIP permet de créer une bande de triangles.
Dans OpenGL, une bande de triangles se définie en définissant les sommets dans l'odre s1, s2, s3,...
Une succession de sommet s1, s2, s3, s4, s5, s6,.. définit une bande de triangles consituée des triangles (s1,s2,s3), (s3,s4,s5), (s4,s3,s2) et (s6,s5,s4)
Le schéma suivant illustre le fait que OpenGL constitue la bande de triangles en prenant les sommets dans le sens des aiguilles d'une montre.



Exemple
Exemple
Le programme suivant crée une bande de 3 triangles à partir de de 5 sommets.

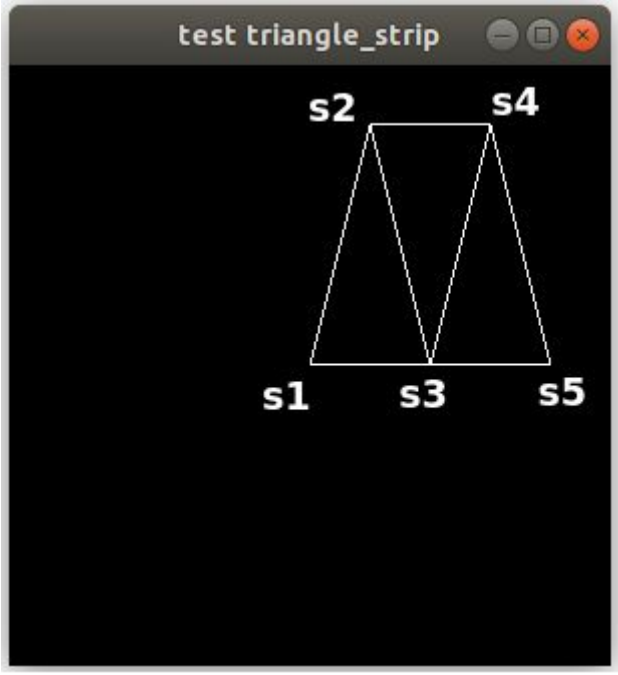
```
/*
* testGL_TRIANGLE_STRIP
* compilation:  g++ test.cpp -o test -lglut -lGLU -lGL
* exécution: ./test
*/

#include <GL/freeglut.h>
void affiche(void)
{
glClear(GL_COLOR_BUFFER_BIT);
glColor3f(0.0, 0.0, 0.0);
glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);

glBegin(GL_TRIANGLE_STRIP);
glVertex3f(0.0, 0.0, 0.0);//s1
glVertex3f(30.0, 120.0, 0.0);//s2
glVertex3f(60.0, 0.0, 0.0);//s3
glVertex3f(90.0, 120.0, 0.0);//s4
glVertex3f(120.0, 0.0, 0.0);//s5
glEnd();
```

```
glFlush();
}
void resize(int w, int h)
{
glViewport(0, 0, w, h);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glOrtho(-150.0, 150.0, -150.0, 150.0, -1.0, 1.0);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
}

// Main routine.
int main(int argc, char **argv)
{
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_SINGLE | GLUT_RGBA);
glutInitWindowSize(300, 300);
glutInitWindowPosition(100, 100);
glutCreateWindow("test triangle_strip");
glutDisplayFunc(affiche);
glutReshapeFunc(resize);
glClearColor(1.0, 1.0, 1.0, 0.0);
glutMainLoop();
}
```



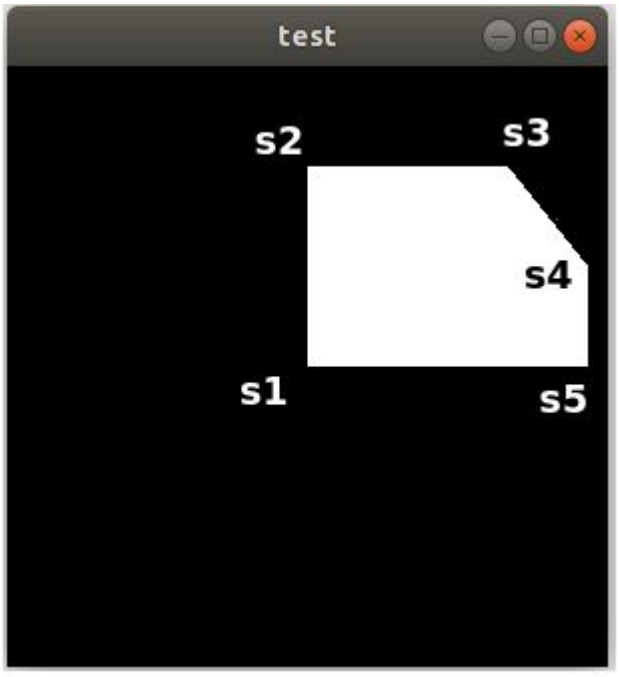
7) Type GL_TRIANGLE_FAN

Le type GL TRIANGLE FAN permet de créer un évantail de triangles

Un évenatail de triangles est par un nombre n de sommets.
Chaque sommet est défini par ses coordonnées x,y et z qui sont ses coordonnées dans l'espace du modèle ; le modèle étant l'objet 3D définit entre les instructions glBegin() et glEnd().
Une succession de sommets s1, s2, s3, s4, s5... défini un éventail de triangles constitué des triangles (s1,s2 ,3), (s1, s3, s4), (s1,s4,s5),...

Exemple
Le programme suivant crée un éventail de 3 triangles à partir de 5 sommets.

```
glBegin(GL_TRIANGLE_FAN);
glVertex3f(0.0, 0.0, 0.0);//s1
glVertex3f(0.0, 100.0, 0.0);//s2
glVertex3f(100.0, 100.0, 0.0);//s3
glVertex3f(140.0, 50.0, 0.0);//s4
glVertex3f(140.0, 0.0, 0.0);//s5
glEnd();
```



8) Type GL_POLYGON

Le type de primitive **GL_POLYGON** permet de créer des polygones. Chaque polygone est définie par n sommets.
Chaque sommet est défini par ses coordonnées x,y et z qui sont ses coordonnées dans l'espace du modèle ; le modèle étant l'objet 3D définit entre les instructions glBegin() et glEnd().
Une succession de sommet s1, s2, s3, s4, s5... définit le polygone (s1, s2, s3, s4, s5,...)

Exemple
Le programme suivant crée un polygone de 5 sommets.

```
glBegin(GL_POLYGON);
glVertex3f(0.0, 0.0, 0.0);//s1
glVertex3f(0.0, 100.0, 0.0);//s2
glVertex3f(50.0, 100.0, 0.0);//s3
glVertex3f(100.0, 50.0, 0.0);//s4
glVertex3f(100.0, 0.0, 0.0);//s5
glEnd();
```