Synthèse d'Image - TD02

Dessins de base : Objets canoniques et transformations

Lors de cette séance, nous aborderons l'un des concepts les plus simples de la modélisation en 2D (et en 3D) : La construction d'objets canoniques.

EXERCICE 01 : Après les points ...

Lors du précédent TP, vous avez appris à dessiner des points à l'aide des instructions suivantes :

```
glBegin(GL_POINTS);
      glVertex2f(x,y);
glEnd();
```

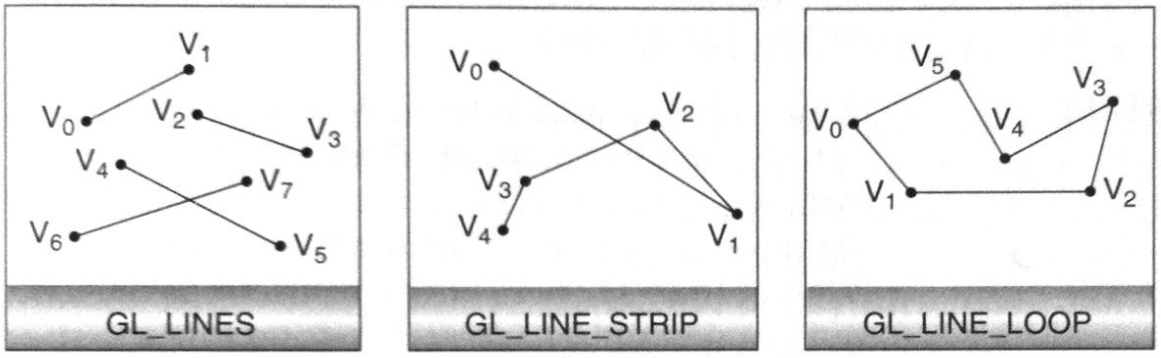
Entre `glBegin` et `glEnd`, il est possible d'utiliser:

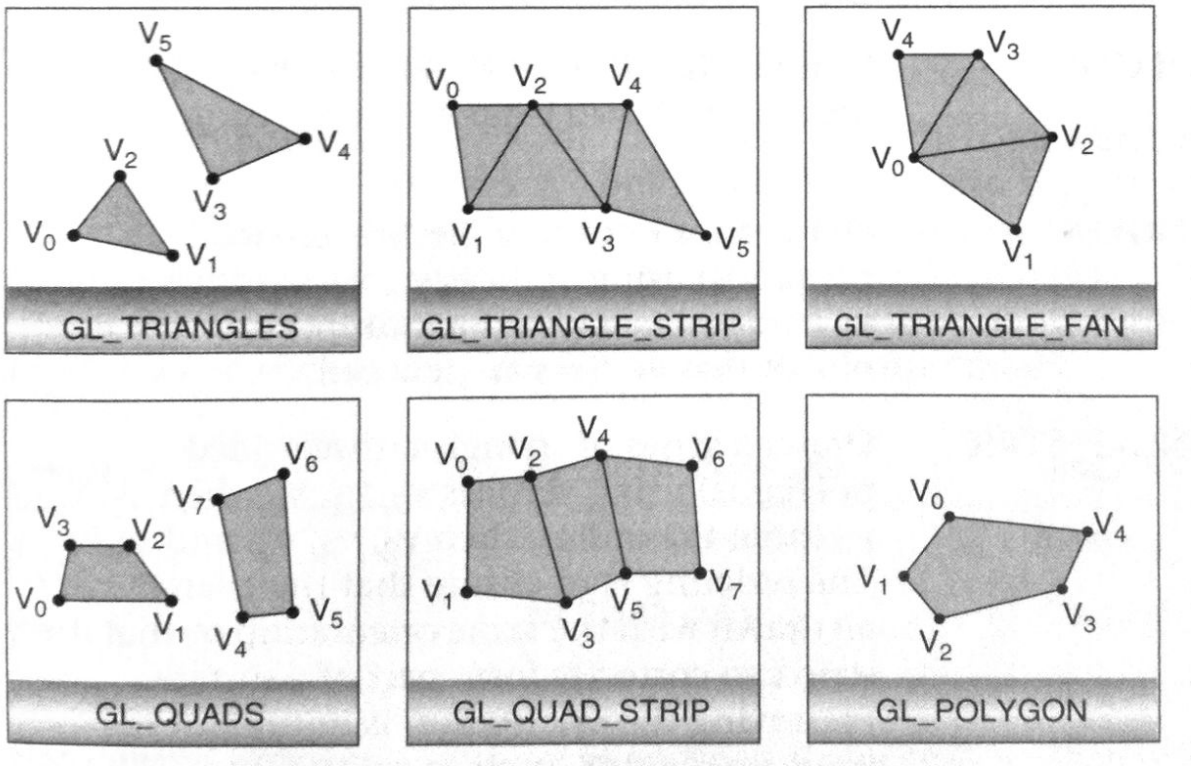
- `glColor3f` (ou autres variantes de `glColor`)
- `glVertex2f` (ou autres variantes de `glVertex`)
- tout ce que vous avez appris en C (boucle, tests, etc.)

Les autres fonctions GL ou SDL ne sont pas acceptées.

L'ensemble des **"vertices"** ("sommets" en Francais) que vous créez entre ces deux instructions forme la liste des sommets qui seront affichés. Par exemple, si vous utilisez `GL_POINTS` en paramètre de `glBegin`, alors chaque appel à `glVertex` dessinera un unique point.

Il existe d'autres valeurs possibles à part `GL_POINTS`.
On appelle ces valeurs des **primitives**, et en voici la liste :





Pour tracer un segment de droite en OpenGL, vous devrez donc utiliser les instructions suivantes :

```
glBegin(GL_LINES);
      glVertex2f(x0,y0); //point de depart
      glVertex2f(x1,y1); //point d'arrive
glEnd();
```

Remarquez qu'il vous faut au minimum 2 points entre `glBegin()` et `glEnd()` pour pouvoir tracer un segment. Pour en tracer un second il vous faudra de nouveau 2 points, et ainsi de suite.

De même, vous aurez besoin au minimum de 3 points pour pouvoir dessiner des triangles ou des polygones et de 4 points pour pouvoir dessiner des carrés.

Note : Dans le cas ou il y aurait un nombre insuffisant de points, la primitive demandée ne sera pas dessinée.

1. **Pour vous échauffer, faites une application qui dessine une ligne brisée par clics successifs dans la fenêtre.** Il sera nécessaire de stocker tous les points cliques et d'afficher entièrement la ligne a chaque dessin de la fenêtre. Vous pouvez utiliser un tableau, ou mieux une liste de structure de points (réutilisez la liste de points de l'exercice 6 du TD01 !).

2. **Faites en sorte que lorsqu'on clique avec le bouton de droite de la souris, le point cliqué devienne le dernier de la ligne qui doit alors boucler sur elle même (le dernier point est lié au premier).**

EXERCICE 02 : Des objets canoniques ...

Votre maîtrise des primitives fraichement acquise, vous allez maintenant pouvoir créer des objets plus complexes. Ces objets seront de taille unitaire, et centrés sur l'origine.

Réalisez les fonctions suivantes permettant chacune de dessiner un objet canonique :

1. **`drawSquare()`**
   pour dessiner un carre de cote 1 centre sur l'origine.

2. **`drawLandmark()`**
   pour dessiner un repère, c'est dire :
   - Un segment de taille 1 et de couleur **rouge** sur l'axe horizontal (axe des x).
   - Un segment de taille 1 et de couleur **vert** sur l'axe vertical (axe des y).

3. **`drawCircle()`**
   pour dessiner un cercle de diamètre 1 centré sur l'origine.
   Pour dessiner le cercle, vous utiliserez une segmentation de ce dernier. C'est a dire une approximation du cercle a l'aide de plusieurs segments de droite. Ce nombre de segments est a définir en tant que constante dans votre fichier d'entête.

4. **Finalement, affichez ces objets canoniques dans votre application.**

Rappel : Tout dessin doit se faire entre `glClear(GL_COLOR_BUFFER_BIT)` et `SDL_GL_SwapBuffers()`.

Bonus : Ajoutez un paramètre `int full` aux fonctions précédentes. Ce paramètre permettra de choisir si l'objet est dessiné rempli ou s'il s'agit d'un contour.

EXERCICE 03 : Transformations ...

Ces objets canoniques sont peu intéressants en l'état. Cependant, OpenGL a la particularité de pouvoir déplacer et/ou déformer les "objets" qu'il dessinera plus tard.

Cela se fait en modifiant la **matrice de transformation** d'OpenGL. Pour modifier cette matrice, il faut utiliser l'instruction suivante :

```
glMatrixMode(GL_MODELVIEW);
```

Cette instruction permet de dire à OpenGL que la **matrice courante** (celle que l'on va modifier juste après) est la matrice de transformation. Il existe d'autres types de matrices comme la matrice de projection `GL_PROJECTION` que vous avez utilisé au TD01.

Pour demander le chargement de la matrice identité, on utilise

```
glLoadIdentity();
```

Cette instruction **modifie la matrice courante** pour la faire devenir une **matrice identité**.

On peut également effectuer une **translation** à l'aide de

```
glTranslatef(x,y,z);
```

où *x, y et z* doivent être des flottants.

On peut demander une **rotation** dans le plan **x0y** a l'aide de

```
glRotatef(alpha, 0.0, 0.0, 1.0);
```

où *alpha* est l'angle de rotation en **degrés**.

Enfin, on peut effectuer un **scale** (un changement d'échelle) à l'aide de

```
glScalef(x,y,z);
```

où *x, y et z* doivent être des flottants.

Note : La matrice de transformation `GL_MODELVIEW` s'appliquera à tous les points que vous dessinerez par la suite.

1. **Modifiez la taille de votre fenêtre virtuelle pour qu'elle fasse 8 unité en x, 6 en y et soit toujours centrée en (0,0).** Pour vous aider, vous pouvez placer des points à différentes coordonnées pour pouvoir vous repérer.

2. **Dessinez un repère dans votre fenêtre.**

3. **Dessinez un cercle orange centré en (1; 2). Attention : vous ne devez pas modifier votre fonction de dessin du cercle.**

4. **Ajoutez un carré rouge sur lequel vous appliquerez une rotation de 45° puis une translation de 2 sur l'axe des x.**

5. **Permutez la rotation et la translation pour dessiner un carré violet supplémentaire. Que se passe-t-il ?**

6. **Dessinez un carré jaune en (0; 0). Puis faites en sorte que lorsque vous cliquez quelque part dans la fenêtre, le carré se recentre à l'endroit cliqué.**

7. **Faites en sorte qu'en faisant un *drag and drop* avec le clic droit de la souris, on effectue une rotation du carré jaune (par rapport a son centre).**

8. **Faites en sorte qu'un cercle bleu se déplace aléatoirement dans la fenêtre et ce en permanence. Vous pourrez vous aider de la fonction `rand()` et le header `time.h`.**