C++
TP n°3

Master Informatique IFI-RIF-MBDS 1ère année
Université de Nice-Sophia Antipolis

1 Installation de libGraph

1.1 Commun

Allez sur la page du TP et récupérez l'archive `uselibgraph.zip`. Un makefile est à disposition pour pouvoir utiliser ces fichiers.

1.2 Linux

Dans un terminal entrez la commande:

```
sudo apt-get install freeglut3-dev
```

Dans le dossier include de l'archive, supprimer le dossier GL.

2 Utilisation de libGraph

Lors de vos utilisation de libGraph, il vous faudra hériter des classes `ControlEngine`, `GameEngine` et `GraphicEngine` pour respectivement redéfinir les callbacks de contrôle, d'idle et graphique.

2.1 ControlEngine

La classe ControlEngine possède 8 méthodes qui pourront être redéfinit pour changer les contrôles de votre application.

```
virtual void KeyboardCallback(unsigned char key,int x, int y);
```

Que l'on pourra "override" pour gérer les pressions sur les touches du clavier.

```
virtual void KeyboardReleaseCallback(unsigned char key,int x, int y);
```

Que l'on pourra "override" pour gérer les relachements sur les touches du clavier.

```
virtual void specialCallback(int key,int x, int y);
```

Que l'on pourra "override" pour gérer les pressions sur les touches spéciales du clavier.

```
virtual void MouseCallback(int button, int state, int x, int y);
```

Que l'on pourra "override" pour gérer les pressions sur les boutons de la souris.

```
virtual void MotionCallback(int x, int y);
```

Que l'on pourra "override" pour gérer les déplacements de la souris lorsqu'un ou plusieurs boutons sont préssés.

```
virtual void PassiveMotionCallback(int x, int y);
```

Que l'on pourra "override" pour gérer les déplacements de la souris lorsque aucun bouton n'est pressé.

```
virtual void MouseEntry(int state);
```

Que l'on pourra "override" pour gérer les entrées et sorties de la souris de la fenêtre.

```
virtual void JoystickCallback(unsigned int buttonMask,
                        int x, int y, int z);
```

Que l'on pourra "override" et qui comme son nom l'indique permettra de gérer une manette.
Si vous avez besoin de renseignements sur certains arguments, rendez-vous sur la documentation officielle de glut, librairie sur laquelle est basée libGraph.

2.2 GameEngine

La classe GameEngine ne possède qu'une méthode, mais c'est une méthode fondamentale. C'est grâce à elle que la totalité de vos jeux seront animé. La méthode :

```
virtual void idle();
```

Que l'on pourra "override" sera appelé par l'application en continue (plus exactement, à chaque fois que l'application n'aura rien à faire).

2.3 GraphicEngine

Dans le moteur graphique, il y à deux méthodes:

```
virtual void Draw();
```

Que l'on pourra "override" et qui contiendra la totalité des appels des primitives de dessins.

```
virtual void reshape(int width, int height);
```

Que l'on pourra "override" et qui sera appelée à chaque redimensionnement de la fenêtre.

2.4 GraphicPrimitives

la classe GraphicPrimitives contient la totalité des méthodes de dessins de la libGraph.

```
static void drawFillRect2D(float x,float y,float width,float height,float r,float g, float b,flo
```

Dessine un rectangle en `(x,y)` de largeur `width` et hauteur `height` et de couleur `r,g,b,a`.

```
static void drawOutlinedRect2D(float x,float y,float width,float height,float r,float g, float b
```

Dessine un rectangle avec juste ses arrètes en `(x,y)` de largeur `width` et hauteur `height` et de couleur `r,g,b,a`.

```
static void drawLine2D(float x1,float y1,float x2,float y2,float r,float g, float b,float a = 1.
```

Dessine une ligne du point `(x1,y1)` au point `(x2,y2)` de couleur `r,g,b,a`

```
static void drawFillTriangle2D(float x1,float y1,float x2,float y2,float x3,float y3,float r,flo
```

Dessine un triangle qui relit les point 1 2 et 3, de couleur `r,g,b,a`

```
static void drawOutlinedPolygone2D(std::vector<float> &x,std::vector<float> &y,float r,float g,
```

Dessine un polygone avec juste ses arrètes (le dernier sommet rejoindra le premier)

```
static void drawFillPolygone2D(std::vector<float> &x,std::vector<float> &y,float r,float g, floa
```

Dessine un polygone remplit (le dernier sommet rejoindra le premier)

```
static void drawText2D(char * str,float x,float y,float r,float g, float b,float a = 1.0f);
```

Ecrit le texte a l'écran.

3 Exemple simple

Dans le dossier uselibgraph, se trouve un exemple d'utilisation de la libGraph, celui-ci se décompose en 3 classes importantes.

3.1 MyControlEngine

Qui va redéfinir:

```
virtual void MouseCallback(int button, int state, int x, int y);
```

Les valeurs de x et y sont les coordonnées par rapport au haut à gauche de l'écran. La valeur `GLUT_LEFT_BUTTON` représente le bouton gauche et `GLUT_DOWN` représente la pression.

3.2 MyGameEngine

Qui va redéfinir:

```
virtual void idle();
```

Dans notre exemple, idle ne fait qu'appeler une méthode de la classe Papillons, qui fera bouger les Papillons.

3.3 MyGraphicEngine

Qui va redéfinir:

```
virtual void Draw();
```

Qui va simplement appeler pour chaque Papillons sa propre méthode draw().

4 Exercice 1

Modifiez l'exemple pour modifier:

- La couleur des papillons.
- Faire en sorte que chaque papillons ai sa propre couleur.
- Afficher à l'écran le nombre de papillons.
- Rajouter une nouvelle espèce, les fourmis, dessiné par un triangle, avec la pointe qui pointe vers la direction actuelle.
- Bonus: lorsqu'un papillon croise une fourmis, le papillon mange la fourmis.

5 Exercice 2

Le but de cette exercice va être de pourvoir gérer un ensemble de case.

- Dessiner un damier à l'écran.
- Lorsque l'on clic sur une case, la case doit changer de couleur.
- Chaque case possède sont propre compteur de clic et l'affiche.

6 Exercice 3 : Base d'un jeu de defense

6.1 1 ligne

Dans un damier de 1 ligne pour 10 colonnes. vous devez donner la possibilité à l'utilisateur de poser un vaisseau (représenté par un triangle) sur une case qui tirera avec une certaine `fréquence` des missiles d'une certaine `puissance` et `vitesse` vers la droite.
Sur cette même ligne mais uniquement du coté droit, apparaitrons des asteroids (représenté par un polygone) qui se déplaceront à une certaine `vitesse` et possédant un certaine nombre de `vie`. Lorsque les asteroids perdent la totalité de leur vie, ils sont détruit.

6.2 Multiligne

Il va falloir maintenant gérer une multitude de lignes, avoir plusieurs sortes de vaisseaux (au moins 3) et plusieurs sortes d'asteroids (au moins 3).