INFO-F-202
Langages de programmation 2
Université libre de Bruxelles
Laboratoire 2

Justin Dallant John Iacono
Yannick Molinghen Alexis Reynouard

28 Septembre 2021

1 Tu devrais avoir ...

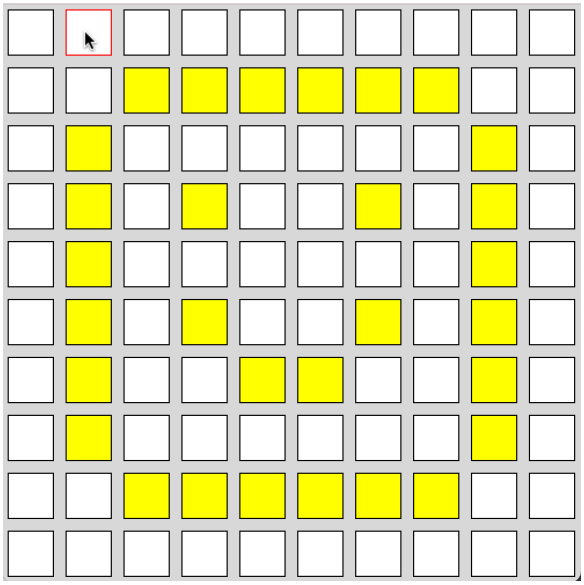
Vous devriez avoir regardé les vidéos suivantes :

1. Le cours : structure, logistique et les notes
2. Pourquoi C++
3. Encapsulation
4. Durée de vie des objets I : variables automatiques

Rappel : Les vidéos que vous devez regarder chaque semaine sont postées sur UV.

2 Objects !

Cette semaine, nous allons commencer la programmation orientée objet. L'objectif final est le suivant :



Lorsque qu'un carré est survolé par la souris, sa bordure devient rouge. À chaque fois que vous cliquez dans un carré, sa couleur bascule entre le blanc et le jaune.

3 Les Classes

Vous allez coder plusieurs classes. Je décris chacune d'elles brièvement avec les méthodes publiques suggérées.

Point Nous l'avons utilisé la semaine dernière. Rien à coder :

```
struct Point {
    int x,y;
};
```

Je vous rappelle que vous pouvez créer une nouvelle variable de point en utilisant `Point p{1,2}` ou un temporaire avec juste le nom de la classe : `Point{1,2}`.

Rectangle Cette classe doit représenter un seul rectangle, avec l'intérieur et la bordure dessinés dans deux couleurs spécifiées. Il sera situé à une position fournie dans le constructeur et ne changera pas de position. La classe aura une méthode de dessin, `draw`, et des méthodes pour changer les couleurs. Enfin, elle aura une méthode `contains`, qui, étant donné un point, renvoie si le point est à l'intérieur du rectangle.

```
class Rectangle{
    // Add things here
public:
    Rectangle(Point center, int w, int h,
        Fl_Color frameColor=FL_BLACK,
        Fl_Color fillColor=FL_WHITE);
    void draw();
    void setFillColor(Fl_Color newFillColor);
    void setFrameColor(Fl_Color newFrameColor);
    bool contains(Point p);
};
```

Cell La classe `Cell` est censée représenter logiquement un rectangle qui est dessiné ainsi que son état. La classe `Rectangle` est destinée à être réutilisée à l'avenir, alors que `Cell` contiendra toute la logique spécifique au comportement du rectangle dans ce programme.

La classe `Cell` aura comme variable d'instance un `Rectangle`, et le gérera en changeant ses couleurs. Ses méthodes seront appelées par la classe `Canvas` à chaque fois que la souris bouge, quand un clic est effectué ou quand il faut la dessiner.

```
class Cell {
   // There should be a rectangle here!
public:
    Cell(Point center, int w, int h);
    void draw();
    void mouseMove(Point mouseLoc);
    void mouseClick(Point mouseLoc);
};
```

Canvas La classe `Canvas` est notre classe de haut niveau. Elle contiendra les `Cells` en tant que variables d'instance et les initialisera. Elle aura des méthodes `draw`, `mouseMove` et `mouseDown` qui sont appelées. La tâche est d'appeler les méthodes correspondantes de chaque `Cell` à partir de chacune des méthodes de `Canvas`.

```
class Canvas{
    // Need to store cells as instance variable(s)
public:
    Canvas();
    void draw();
    void mouseMove(Point mouseLoc);
    void mouseClick(Point mouseLoc);
    void keyPressed(int keyCode) {exit(0);}
};
```

MainWindow N'y touchez pas ! Elle crée une instance unique de notre classe `Canvas` et assure que `draw` soit appelé 60 fois par seconde sur notre classe `Canvas`, ainsi que les appels `mouseMove`, `mouseClick` et `keyPressed` de `Canvas` lorsque ces événements se produisent.

Sommaire Nous pouvons voir le besoin de dessiner via `draw` et de gérer des événements tels que le déplacement ou le clic de la souris comme des messages. Ces messages sont transmis de `MainWindow` à `Canvas`. À chaque étape, une classe peut gérer elle-même le message ou le transmettre à d'autres classes. Par exemple, pour dessiner, lorsque `Canvas` a besoin de dessiner, il demande à chaque `Cell` de dessiner, ce qui demande à chaque `Rectangle` de dessiner. En revanche, les messages de la souris sont transmis à `Cell`, qui les traitera et ne les transmettra pas à `Rectangle`. Cette gestion hiérarchique des événements est typique de la programmation graphique orientée objet.

4 Commencer

4.1 Objectif 1 : Dessiner un seul rectangle

Le code initial qui vous est fourni a un `Canvas` qui a une seule `Cell c`, qu'il construit à l'emplacement 250 250 et à la taille 100 100. Les méthodes `draw`, `mousemove` et `mouseClick` du `Canvas` appellent simplement les mêmes méthodes sur `c`. La classe `Cell` a une seule instance de `Rectangle` qu'elle construit avec l'emplacement et la taille donnés, intérieur blanc et extérieur noir. La méthode `draw` de `Cell` appelle la méthode `draw` de rectangle.

Votre première tâche consiste à faire fonctionner la classe `Rectangle`. Plus précisément, `draw` doit dessiner le `Rectangle` spécifié dans l'appel au constructeur. Vous devrez ajouter des variables d'instance à la classe `Rectangle`.

Aide : Il existe des `Fl_Colors` prédéfinies comme `FL_BLACK`, `FL_WHITE`...

4.2 Objectif 2 : Changer le contour du rectangle en rouge lorsque la souris le survole

Ici, vous devez implémenter le reste des méthodes de `Rectangle`, en particulier `setFrameColor` et `contains`.

Ensuite, écrivez du code dans `mouseMove` qui vérifiera si la souris est à l'intérieur du `Rectangle r` (en utilisant `contains`), puis en changeant la couleur à l'aide de `setFrameColor`.

4.3 Objectif 3 : Basculer entre le blanc et le jaune lors d'un clic

Ajoutez du code à `mouseClick` dans la classe `Cell` pour changer la couleur de l'intérieur du rectangle `r` si on clique à l'intérieur. La cellule devra "se souvenir" du fait qu'elle est actuellement activée ou désactivée.

4.4 Objectif 4 : Passer à une grille de cellules dans `Canvas`

Maintenant, changez `Canvas` pour avoir plusieurs cellules au lieu d'une seule. Dans la démo, il y a 100 cellules disposées dans une grille de 10 par 10. Dans mon code, je les stocke dans un seul `vector<Cell>` et les ajoute au vecteur en utilisant `push_back`. Bien sûr, les méthodes `draw` et les méthodes de la souris doivent maintenant être appelées sur TOUTES les cellules.

Si vous avez correctement codé `Cell` et `Rectangle`, vous n'aurez pas du tout besoin de les modifier à cette étape.