Rapport du projet Starlight

Florian Knop (39310) - Gatien Bovyn (39189)

17 avril 2015

Table des matières

Introduction

Ce document vise à présenter le travail d'analyse et de programmation effectué lors de la réalisation du projet du laboratoire Langage C++ : Starlight.

Ce projet a été réalisé en binôme par Florian Knop, matricule 39310 groupe 2G13, et Gatien Bovyn, matricule 39189 groupe 2G11.

Le programme à concevoir consiste en une implémentation du modèle et d'une interface graphique du jeu baptisé Starlight, puzzle à 2 dimensions basé sur la lumière.

Ce projet a été compilé principalement avec g++ (version 4.8.2 ou supérieure) sous la distribution Linux Ubuntu (ou une de ses dérivée). La version du framework Qt utilisée est la 5.0.2 ou supérieure. Ce projet a été fait sous QtCreator, IDE OpenSource en version 2.8.1 ou supérieure.

Sections

Conventions de nommages utilisées

Dans cette section, nous présenterons les différentes conventions utilisées lors de ce projet. Nous avons décidé de reprendre certaines conventions utilisées par la STL tout en gardant d'autres conventions.

De manière globale, tous les noms de variables, classes, fichiers, etc. sont en anglais.

Nommage des fichiers

Les noms de fichiers sont entièrement en minuscules et possèdent les extensions .h pour les headers et .cpp pour les fichiers sources.

- **nomfichier.h**
- **nomfichier.cpp**

Classes

Les noms des classes commencent par une majuscule à chaque mot. Cette convention est également valable pour les noms d'énumérations et structures.

- **Classe**
- **NomClasse**
- **NomStruct**
- **NomEnumeration**

Variables

Variables de classe

Les noms des variables de classes sont en minuscules, les mots sont séparés par un underscore et le nom est suffixé par un underscore.

— **variable_**
— **nom_variable_**

Cette convention est également valable pour les variables d'une structure.

Variables locales

Les noms des variables locales respectent les mêmes conventions que les variables de classe, à la seule exception qu'ils ne sont pas suffixés par un underscore.

— **variable**
— **nom_variable**

Constantes

Les noms des constantes sont entièrement en majuscules et les mots sont séparés par des underscores.

— **CONSTANTE**
— **NOM_CONSTANTE**

Méthodes

D'une manière générale, les noms de méthodes sont en minuscules et séparés par des underscores.

Getters

Le nom d'un getter (accesseur en lecture de variable de classe/structure) est égal au nom de la variable de classe sans l'underscore final.

— Variable : **variable_** - Getter : **variable()**
— Variable : **nom_var_** - Getter : **nom_var()**

Setters

Le nom d'un setter (accesseur en écriture de variable de classe/structure) est égal au nom de la variable sans l'underscore final préfixé de `set`.

— Variable : **variable_** - Setter : **set_variable()**
— Variable : **nom_var_** - Setter : **set_nom_var()**

Autres méthodes

Les autres méthodes possèdent les mêmes conventions que celles énoncées ci-dessus.

— **methode()**
— **nom_methode()**

Éléments d'une énumération

Les éléments d'une énumération suivent les mêmes conventions que les conventions des constantes énoncées ci-dessus.

Présentation des différentes classes

Dans cette section, nous allons décrire les différentes classes composants ce projet. L'implémentation du projet est divisée entre la partie modèle et la partie vue. Elle est également basée sur le design pattern Observeur / Observé comme demandé dans les consignes.

Modèle

Dans cette section, nous allons décrire les différentes classes du modèle (classes métiers). Un squelette de classes a été fourni par Monsieur Absil. Ce squelette contenait les fichiers suivants : 'point.h, source.h, dest.h, nuke.h, wall.h, crystal.h, lens.h, mirror.h, ray.h, level.h'. Nous avons décidé de modifier ce squelette tout en gardant la structure générale.

Line

Cette classe représente une droite, elle possède un point et un angle. À l'aide de ces informations, on peut trouver n'importe quel point de la droite en ayant la distance entre le point d'origine et le point d'arrivée.

Dans les méthodes intersects(...) de la classe Line, un passage par pointeur de pointeur est fait car un pointeur est passé par valeur et lors de l'initialisation il ne pointera donc pas la même adresse mémoire que le pointeur d'origine. Si on passait donc par simple pointeur, notre pointeur copié aurait une bonne zone mémoire mais notre pointeur d'origine resterait à *nullptr*.

LineSegment

Cette classe représente un segment de droite, elle possède deux points qui représentent les extrémités du segment.

Ellipse

Cette classe représente une conique de forme elliptique. C'est-à-dire une ellipse ou un cercle.

Rectangle

Cette classe représente une forme géométrique rectangulaire. Elle possède un point supérieur gauche ainsi qu'une longueur et hauteur nous permettant de retrouver facilement les autres extrémités de la forme.

Cette classe sert à représenter les objets Source et Destination.

Level

Cette classe représente la classe principale du jeu, elle gère toute la logique métier du jeu. Elle permet de calculer la trajectoire du rayon lumineux.

Vue

L'interface graphique a été réalisée en 'Qt' à la main.

Les classes composant la partie vue de l'application sont :

DestinationView

Description Classe modélisant la destination à atteindre par le rayon émis depuis la source pour gagner la partie.

MapView

Description Classe représentant le plateau de jeu, où tous les éléments sont disposés et affichés.

MirrorView

Description Classe représentant un miroir sur lequel un rayon peut être réfléchi.

NukeView

Description Classe représentant une bombe, élément explosif du plateau qui fait perdre la partie si touché par un rayon lumineux (RayView).

SourceView

Description Classe représentant une source lumineuse sur le plateau de jeu. Elle dispose de 2 états, allumée ou éteinte. Passer d'un état à l'autre change l'image la représentant et produit un son d'interrupteur.

WallView

Description Classe représentant un mur, élément visuel sur lequel le rayon est stoppé.

Conclusion

Bibliographie

Annexes

Démarches mathématiques

Trouver l'intersection entre deux droites

$$D \equiv y = ax + b \tag{1}$$

$$D \equiv x = k \tag{2}$$

(1) représente une droite non verticale. (2) représente une droite verticale où k est une valeur quelquonque sur l'axe des x.

Dans un premier temps, il faut tester que la pente a est différente. Si cette dernière est égale, il n'y a pas d'intersections. En effet, si les pentes sont les mêmes, cela signifie que les droites sont soit parallèles soit égales. Dans le cas de droites égales, on considère qu'il n'y a pas d'intersections, bien qu'en réalité il y en a une infinité.

Si la pente est différente, l'intersection entre deux droites est assez simple, il suffit d'injecter une variable d'une des deux équations dans l'autre. Dans le cadre du projet, il faut evidemment faire attention aux droites verticales.

Le cas d'une droite verticale :

On injecte (2) dans (1).

$$D \equiv y = ak + b \tag{3}$$

Avec (3) on obtient y du point dont le $x = k$ pour une droite verticale (2).

Pour deux droites non verticales on injecte le y d'une équation de type (1) dans le y d'une autre équation de typ (1).

$$aremplir \tag{4}$$

$x = \frac{(y-b)}{a}$ pour une droite non verticale dont le y est celui obtenu plus tôt (1).

Trouver l'intersection entre une droite et un segment

Trouver le(s) intersections entre une ellipse et une droite

La formule d'une ellipse est :

$$E \equiv \frac{(x - x1)^2}{a^2} + \frac{(y - y1)^2}{b^2} = 1$$

où $x1$ et $y1$ sont respectivement les coordonnées x et y du centre de l'ellipse.

où a et b sont respectement les rayons de l'axe x et y.

Pour trouver une intersection entre une ellipse et une droite, il faut égaler deux variables identiques :

On doit donc remplacer la variable x ou y de la droite dans l'équation de l'ellipse.

Le cas de la droite verticale :

Dans ce cas la, il n'y a pas de choix, il faut remplacer x dans l'équation de l'ellipse. Nous avons également décidé de refactoriser l'équation en prenant le PPCM (Plus petit commun multiple) de $a^2 * b^2$ que nous apellerons ici lcm pour Least Commun Multiple. Ce choix a été fait pour éviter les overflows lorsque de nombres trop grands sont mis au carré et multipliés. Bien que dans notre cas, on nous avons rarement des nombre pouvant obtenir un tel résultat.

$$E \equiv (lcmy \cdot (k - x1)^2) + ((y - y1)^2 \cdot lcmx) = lcm$$

où $lcmy$ est le facteur par lequel il faut multiplier b^2 (rayon y au carré) pour obtenir lcm,
où $lcmx$ est le facteur par lequel il faut multiplier a^2 (rayon x au carré) pour obtenir lcm.

$$E \equiv lcmy \cdot (k - x1)^2 + (y^2 + y1^2 - 2 \cdot y1 \cdot y) \cdot lcmx = lcm$$
$$E \equiv lcmy \cdot (k - x1)^2 + lcmx \cdot y^2 + lcmx \cdot y1^2 - 2 \cdot lcmx \cdot y1 \cdot y - lcm = 0$$

Avec ceci, il reste plus qu'à résoudre l'équation du second degré avec

$$\rho = b^2 - 4ac$$

où

$$a = lcmx$$
$$b = 2 \cdot lcmx \cdot y1 \cdot y$$
$$c = (lcmy \cdot (k - x1)^2) + (lcmx \cdot y1^2) - lcm$$

Le nombre d'intersections est différent selon la valeur de ρ.

$$n = \begin{cases} 0 & \text{si } \rho < 0 \\ 1 & \text{si } \rho = 0 \\ 2 & \text{si } \rho > 0 \end{cases}$$

$$y = \frac{-b}{2a}$$

$$y1 = \frac{(-b + \sqrt{\rho})}{2a}$$
$$y2 = \frac{(-b - \sqrt{\rho})}{2a}$$

On a donc le(s) y du/des points d'intersections, et le x vaut k (de l'équation de départ).

Le cas de la droite non verticale :

C'est le même principe que le cas de la droite verticale sauf que pour trouver la deuxième variable finale il faudra remplacer la variable trouvée dans l'équation de la droite.

Trouver le(s) intersections entre une ellipse et un segment

Il s'agit du même principe que les intersections droite/segment. Il faut vérifier les intersections entre les ellipses et un segment transformé en droite et ensuite vérifier si les points d'intersections se trouvent sur le segment.