Compte Rendu TP1

Hadrien Marquez

08/05/2017

Installation

```
In the project directory :
Run 'cmake CMakeLists.txt' to create makefile.
Run 'make' to build project.
Run './bin/toto' to start the application.
```

Table des matières

1 Consigne

Ce TP avait pour objectif de nous faire calculer une approximation de pi. Pour cela on utilise la méthode de Monte Carlo. Ce calcul est avant tout une excuse pour nous permettre de tester la programmation orienté objet.

La méthode de Monte Carlo consiste en un cercle et une collection de point aux coordonnées aléatoires. Ces coordonnées aléatoires, sont tout de même limité par le carré de côté égale au diamètre de notre cercle. On vérifie ensuite pour chaque point s'il est à l'intérieur du cercle ou non. Dans le cas où le point est dans le cercle, on incrémente un compteur. Pi est donné par l'opération suivante : le compteur multiplié par 4 et divisé par le nombre total de point généré.

En figure 1 le résultat de notre programme :

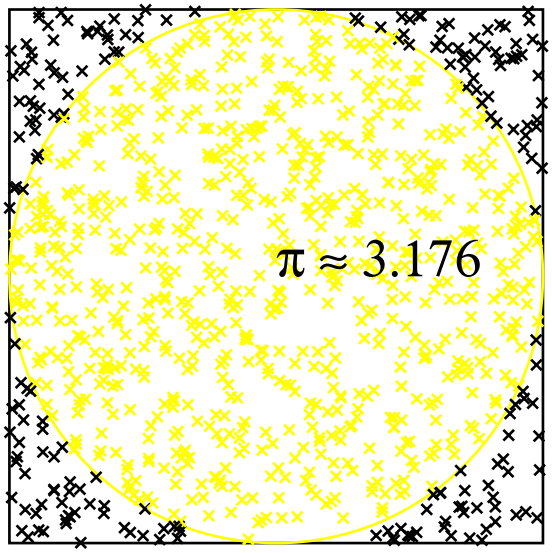

FIGURE 1 – Résultat

2 Structure du programme

En lisant la consigne, j'ai pensé nécessaire de créer une classe pour un cercle et une classe pour un point. Le carré n'a pas une importance capitale dans le calcul. Il ne sert que de limite aux points. Je n'ai pas vu l'intérêt de créer un objet Carré. A ces deux classes s'ajoute la boucle principale du programme. En plus, j'ai voulu essayé les exceptions dans ce petit programme. J'ai donc ajouté une classe héritant de l'exception générique.

2.1 Cercle

La classe Cercle représente un cercle basique composé d'un point central et d'un rayon. Cette classe posséde des méthodes lui permettant de calculer pi. La première ne fait que le calcul. La seconde permet en plus d'avoir en sortie un fichier eps avec le cercle, le carré et les points générés dessinés. Pour simplifier l'utilisation de la classe, des accesseurs (lecture uniquement) ont été implémentés ainsi qu'une surcharge de l'opérateur de flux de sortie ' « '.

2.2 Point

La classe Point, est un point 2-D composé de deux coordonées (x,y). Tout comme le cercle, on a cherché à simplifier au maximum l'utilisation de la classe. Elle est dotée d'accesseurs aux variables membres, la surcharge des opérateurs de flux '«' et '»' et la surcharge des opérateurs '+=' et '*='. Cette dernière surcharge permet en externe de la classe de surcharger les opérateurs d'addition et de multiplication afin de facilement les utilisers en calculs.

2.3 Error

La classe Error hérite de la classe générique exception, inclut grâce à `#include <exception>`. Notre erreur est caractérisé par un numéro, une description du problème et un niveau d'erreur. Elle utilise le polymorphisme sur la méthode what() héritée de exception, celle-ci renvoyant la description du problème.

3 Tests

3.1 $rayon = 0$

Le calcul de π renvoit 0. Le dessin eps quant à lui refuse de s'ouvrir.

3.2 $x! = y$, rayon petit

Plus x et y sont différents et plus pi se rapproche de 0. On a en effet pour $x = 0$, $y = 100$ et un $rayon = 50$ une approximation de $\pi = 0$. Pour regagner en précision, il vaut soit réduire l'écart entre les valeurs de x et de y, soit augmenter le rayon.

3.3 $x! = 0$, $y! = 0$

Sur le dessin, un centre de cercle différent de (0,0) produira un dessin décalé. Voir figure 2.

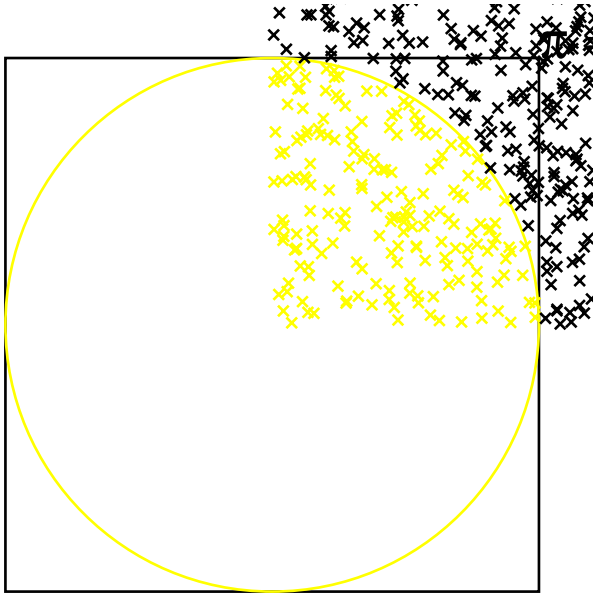

FIGURE 2 – Résultat décalé

3.4 x et y égale à une lettre

Le programme s'emballe et calcule en continue π. Impossible de quitter avec le menu. L'eps de sortie s'ouvre, mais n'affiche rien.