Programmation Orientée Objets L3 MIAGE – INFO M1 IMDS

TD 3 : programmation d'une tortue

Tortue graphique

On a programmé une tortue graphique à la LOGO. Une tortue a une position et un cap de déplacement (exprimé en degrés à partir de l'horizontale). Une tortue a aussi un « pinceau » qui peut être soit levé, soit baissé.

La tortue est initialement placée en (0,0) et orientée avec un angle de 0 degré (horizontale), pinceau baissé. On peut aussi l'initialiser avec une position et une direction données.

Étant donnée une tortue, on peut : lever ou baisser son pinceau (`raisePen()`, `lowerPen()`); changer sa position ou son cap (`setPosition(pos)`, `setHeading(cap)`) ; la faire tourner à gauche ou à droite d'un angle *a* par rapport à son cap (`turnLeft(a)`, `turnRight(a)`); la faire avancer d'une distance *d* suivant son cap (`forward(d)`). Si le pinceau est baissé, alors la tortue laisse une trace sur la fenêtre graphique.

Soit donc la classe `turtle` suivante représentant une tortue :

```
namespace logo {
class turtle {
public :
  turtle();
  turtle(const geom::point& pos, int direction);
  int          heading() const;
  geom::point  position()  const;
  bool         isPenUp()   const;
  bool         isPenDown() const;
  void forward(int step);
  void setPosition(const geom::point& pos);
  void turnLeft(int angle);
  void turnRight(int angle);
  void setHeading(int head);
  void raisePen();
  void lowerPen();
private :
  geom::point d_pos;
  int         d_heading;
  bool        d_pendown;
};
}
```

Tracé d'un polygone régulier

Pour tracer un polygone régulier à *n* côtés il faut

- faire *n* fois
  - avancer la tortue de la longueur du côté
  - tourner à gauche la tortue d'un angle de 360/*n* degrés
- faire tourner à gauche la tortue d'un angle de 360 modulo *n* degrés

Écrivez la classe `regularPolygon` dont le constructeur prend en paramètres le nombre de côtés et la longueur d'un côté du polygone régulier et dont la méthode `run(t)` fait tracer à la tortue `t` le polygone régulier.

Tortue étendue

On veut maintenant améliorer la tortue en lui permettant de reculer d'une distance *d* (`backward`) et de faire un demi-tour (`uTurn`). On veut aussi que la tortue garde en mémoire sa position initiale et qu'on puisse l'y ramener (`home`). On veut enfin que la tortue aie une distance de déplacement par défaut que l'on puisse lire (`defaultStep`) et changer (`setDefaultStep`) et qu'on puisse la faire directement avancer (`forward`) ou reculer (`backward`) de cette distance de déplacement par défaut.

En réutilisant intelligemment la tortue, **écrivez** (dans l'espace de nom `logo`) la classe `xTurtle` représentant une tortue étendue contenant ces améliorations.

Écrivez la classe `regularJewel` dont le constructeur prend en paramètres un nombre de côtés et une longueur de côté du polygone régulier et dont la méthode `run(t)` fait tracer à la tortue étendue `t` le « joyau » composé de deux polygones réguliers opposés.

Programmation d'une tortue

Un programme de tortue est une suite de caractères indiquant des actions à effectuer sur la tortue :

- `'F'` = avancer pinceau baissé ;
- `'f'` = avancer pinceau levé ;
- `'+'` = tourner à gauche ;
- `'-'` = tourner à droite.

Ces ordres doivent être exécutés pour un pas, un angle gauche et un angle droit donnés.

Écrivez la classe `progTurtle` représentant un programme de tortue. On peut :

- créer un programme à partir d'une chaîne de caractères, du pas (10 par défaut) et des angles gauche et droit (90º par défaut) ;
- changer les valeurs du pas, de l'angle gauche et de l'angle droit ;
- changer la chaîne de caractères du programme ;
- faire exécuter le programme à une tortue ;

Écrivez une fonction `testProgramme(p,t)` qui affiche à l'écran `"debut de programme"` suivi du programme, applique le programme `p` à la tortue `t` et enfin affiche à l'écran `"fin de programme"`.

Programmes utilisateurs et programmes particuliers

Programmes utilisateurs

On veut pouvoir faire des programmes particuliers, comme par exemple un programme qui permet de tracer un polygone régulier (`polyProg`). Dans ce cas ces programmes sont calculés par la classe elle-même et ils ne doivent pas pouvoir être modifiés directement par l'utilisateur comme c'est le cas actuellement. De plus les valeurs de pas et d'angles peuvent être aussi calculées par le programme et ne doivent pas non plus être directement modifiables.

Pour cela il faut que la classe `progTurtle` ne permette pas à l'utilisateur de modifier lui-même le programme. On dérive alors la classe `userProgTurtle` qui elle permet à l'utilisateur de modifier le programme, et la classe `polyProg` est dérivée de `progTurtle`.

Modifiez la classe `progTurtle` pour interdire à l'utilisateur de modifier les données du programme. **Écrivez** alors la classe `userProgTurtle`, dérivée de `progTurtle`, qui elle doit autoriser l'utilisateur à modifier les données du programme. **Modifiez** alors le programme de test pour demander un `userProgTurtle` à l'utilisateur, et le passer ensuite à la fonction `testprog(p,t)`.

Programme de tracé d'un polygone régulier

Écrivez la classe `polyProg` qui permet de tracer un polygone régulier à *n* côtés. Le programme correspondant est donc `"F+"` répété *n* fois terminé par `"-"`, avec *angle gauche*=360/*n* et *angle droit*=360 modulo *n*. **Testez** ce programme en le passant à `testprog(p,t)`.