IUT de Paris Descartes – Base de la Programmation Objet

Travaux Dirigés – Sujet n°5

Objectifs

- Choisir une structure de donnée adaptée à un problème
- Être sensibilisé au problème de la redondance de données

Une animation

On veut réaliser une animation simple dans laquelle des éléments (représentés par des caractères) se déplacent de case en case sur un terrain de taille finie.

Les éléments ont une position et une direction initiale. A chaque pas de l'animation, ils changent de position en fonction de la direction. Ce déplacement n'est réalisé que s'il ne provoque pas de sortie du terrain. Dans le cas contraire, la direction est changée de manière à simuler un rebond sur un mur. Les collisions entre les éléments ne sont pas prises en compte et en conséquence plusieurs éléments peuvent être simultanément présents à un même endroit.

L'objectif de la séance est de concevoir la structure de l'application. Plusieurs solutions sont envisageables. On part sur la base de deux classes : une pour les éléments et une pour le terrain. Une ébauche de ces deux classes est donnée en annexe.

1. La classe `Terrain` doit stocker les éléments qui y sont présents. Proposez une structure de données.
2. Complétez la définition des classes en précisant les données et les prototypes des méthodes en fonction des choix réalisés ci-dessus. Faites en sorte qu'aucune donnée ne soit redondante. Notez bien que les prototypes du constructeur de la classe `Elément` et la méthode `Terrain.ajoute` peuvent être complétés si nécessaire.
3. Écrivez un court programme réalisant une animation avec deux éléments.
4. Implémentez les méthodes de la classe `Terrain`.

Annexe

```
public class Elément {
  private char id;     // identificateur de l'élément
  private int dx, dy; // déplacement relatif
  ...                 // à compléter

  public Elément(char id, int dx, int dy, ...) { // à compléter
    this.id = id;
    this.dx = dx;
    this.dy = dy;
    ...               // à compléter
  }

  public char getId() {
    return id;
  }
  ... // à compléter
}
```

```
public class Terrain {
  private int largeur, hauteur;
  // à compléter

  // construit un terrain vide
  public Terrain(int largeur, int hauteur) {
    assert(hauteur > 0 && largeur > 0);
    this.largeur = largeur;
    this.hauteur = hauteur;
    ... // à compléter
  }

  // ajoute un élément au terrain
  public void ajoute(Elément e, ...) { // à compléter
    ... // à compléter
  }

  // simule d déplacements des éléments du terrain
  public void simule(int d) {
    ... // à compléter
  }

  // retourne une représentation textuelle du terrain
  public String toString() {
    ... // à compléter
  }
}
```