École de technologie supérieure

Pierre Bélisle, Francis Bourdeau et Mathieu Nayrolles

INF-111 Travail Pratique #1

Travail individuel
Robot Ball

Remise spécifiée en classe

1. Objectifs

Ce travail a pour objectif de mettre en pratique les connaissances de base en programmation à l'aide du langage Java. Il vise la compréhension des concepts suivants :

- Types primitifs (entier, réel, booléen, caractère).
- Opérateurs sur les types de bases (+, -, *, /, %, …).
- Types primitifs composés (chaînes de caractères, enregistrement et tableaux).
- Références (ou adresses).
- Structures de contrôle (sélection et itération).
- Structures de programme (bloc principal et sous-programmes).
- Passage de paramètres par valeur, paramètre formel et paramètre effectif.
- Entrées (clavier) et sorties (écran).
- Normes de programmation.

De plus, ce travail permet de travailler avec une approche modulaire, grâce à l'utilisation de modules utilitaires.

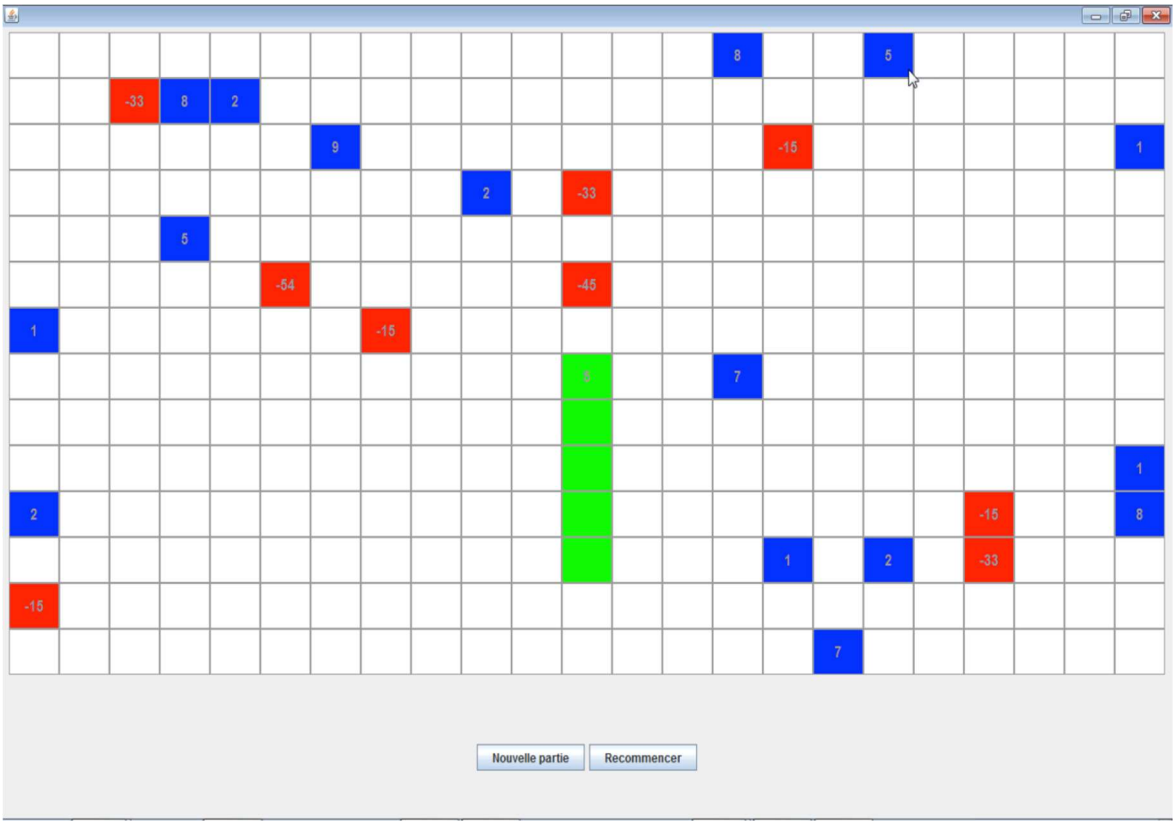
2. Mise en contexte

Ce travail est inspiré d'un jeu appelé « Snake Vs Blocks ». Un robot doit amasser les balles d'énergie qu'il retrouve sur son chemin. Les balles amassées s'ajoutent à une chaîne à l'arrière du robot.

Le robot doit ensuite livrer ces balles à des centres. Chaque centre a un besoin connu en énergie. Lorsque le robot arrive à un centre, il lui donne les balles requises par le centre.

La partie se termine si le robot n'a plus de balle à sa chaîne ou que tous les centres ont été servis.

Des statistiques sont affichées à la fin d'une partie. Dans l'image suivante, le robot et sa chaîne sont verts, les balles sont bleues et les centres sont rouges.



Votre programme doit fonctionner selon les spécifications décrites dans le document qui suit. D'autres détails du jeu vous seront donnés plus loin.

3. Description des modules fournies

Un ensemble de modules déjà implémentés sont fournis. Ces modules utilitaires regroupent un ensemble de sous-programmes ou décrivent un nouvel enregistrement.

Constantes.java

Ce module contient des déclarations de constantes nécessaires à la réalisation de votre projet.

Coordonnee.java

Ce module contient la déclaration d'un enregistrement qui regroupe les coordonnées d'une case présente dans différents jeux de grille. La définition contient deux champs publics qui définissent l'indice de ligne et l'indice de colonne d'une case dans la grille.

Statistiques

Ce module contient la déclaration d'un enregistrement qui regroupe les statistiques à maintenir. On y conserve la plus longue chaine de balles d'énergie conservée par le robot durant une partie, le pointage d'une partie et le pointage de la meilleure partie.

Robot

Ce module retient le nombre de balles et leur position dans la chaîne de balles du robot. La première position du tableau est la position du robot.

UtilitaireFonctions

Ce module contient une fonction pour générer un nombre aléatoire dans un intervalle entier.

GrilleGui

Ce module contient le code pour représenter une grille de jeu rectangulaire de dimensions limitées par MAX_LIGNES et MAX_COLONNES.

Il permet aussi d'obtenir s'il y eu un clic de bouton de souris, la position du clic, la position de la souris, en tout temps, et de modifier le contenu de la case (couleur et texte).

Il est aussi possible d'ajouter des boutons de menu. Ces boutons sont créés en bas de la fenêtre à partir d'un tableau de ***String*** fourni au moment de l'instanciation (voir le sous-programme "GrilleGui").

Voici une brève description des sous-programmes qui s'y trouvent. Consultez les déclarations (spécifications) dans le code pour les détails sur les paramètres. Vous ne devez pas modifier le code de ces sous-programmes.

- **GrilleGui** : crée une grille selon les paramètres reçus. Pour chaque chaîne contenue dans le paramètre "options" un bouton est créé. Un référence "null" transmise au paramètre "option" signifie qu'aucun bouton ne sera créé. Dès qu'on clique sur un bouton, la fonction optionMenuEstCliquee retourne **true** et getOptionMenuClique retourne la chaine de caractère identifiant le bouton.
- **optionMenuEstCliquee** : Retourne ***true*** si un des boutons de menu a été cliqué depuis le dernier appel.
- **getOptionMenuClique** : Retourne la chaine de caractère qui identifie le dernier bouton cliqué et retourne ***null*** autrement.
- **getValeur** : Retourne la valeur contenue dans une case de la grille.

- **setValeur** : Permet de modifier la valeur d'une case de la grille
- **getNbLignes** : Retourne le nombre de lignes de la grille
- **getNbColonnes** : Retourne le nombre de colonnes de la grille.
- **setCouleurFond** : Permet de changer la couleur de fond d'une case. Vous noterez dans le code que le premier paramètre est la coordonnée **y** et vient **x** ensuite (équivalent de ligne, colonne).
- **setCouleurTexte** : Permet de changer la couleur de texte d'une case. Vous noterez dans le code que le premier paramètre est la coordonnée **y** et vient **x** ensuite (équivalent de ligne, colonne).
- **getDernierTouche :** Retourne la dernière touche qui a été appuyée entre TOUCHE_GAUCHE, TOUCHE_HAUT ou TOUCHE_DROITE.
- **pause :** Effectue une pause en millisecondes.

Vous n'avez pas à comprendre tout le code que contient ce module. Vous devez être en mesure d'utiliser les sous-programmes dont vous avez besoin. Ce code n'est pas un exemple à utiliser pour créer des applications graphiques de type GUI. Ici, ce module sert à vous offrir un environnement de jeu plus agréable que le mode console. Nous verrons les bonnes pratiques de création de GUI en temps et lieu.

DemarrerRobotBall

Ce module contient la boucle principale dans la procédure principale (main) qui permet de lancer votre programme. L'algorithme est d'utiliser les différents modules pour effectuer les tâches. Cette procédure est complète et vous est fournie.

4. Description du module à compléter

Le module "UtilitaireJeu" contient les sous-programmes responsables de créer et gérer la grille de jeux contenant les balles d'énergies et les centres de distributions.

Les principales fonctionnalités de ce module sont décrites dans cette section. Toutefois, l'implémentation de ces fonctionnalités nécessite la création d'autres sous-programmes privés afin d'implémenter un module découpé adéquatement.

obtenirNouvelleGrille

Cette procédure publique reçoit :

- un tableau 2D d'entier représentant la grille de jeu.
- le nombre de lignes dans la grille de jeu
- le nombre de colonne dans la grille de jeu

La procédure ajoute aléatoirement des obstacles dans la grille en traitant les lignes une à la fois. Pour chaque ligne, la procédure place une quantité aléatoire de balles d'énergie et une quantité aléatoire de centres de distribution. Sur une ligne, le nombre d'instances d'un type d'obstacle doit être compris entre 0 et le tiers du nombre de colonnes.

Les balles d'énergie sont représentées par les valeurs : 1, 2, 5, 7, 8, 9. Les centres de distributions sont représentés par les valeurs : -5, -15, -33, -45, -54.

Une copie de la grille est conservée dans une variable statique afin de simplifier le redémarrage d'une partie.

afficherGrille

Cette procédure publique reçoit :

- l'interface graphique dans lequel la grille doit être affichée.
- la grille à afficher.

La procédure parcourt la grille pour afficher le contenu de chacune de ces cases dans le GUI. Les cases vides sont représentées par un espace dans une case blanche. Le pointage des balles d'énergies est affiché dans une case rouge alors que le pointage des centres de distribution est affiché dans une case bleue.

afficherRobot

Cette procédure publique reçoit :

- l'interface graphique dans lequel le robot doit être affiché.
- le robot à afficher.

La procédure parcourt toutes les balles trainées par le robot. Grâce aux coordonnées contenues dans chaque balle, elle dessine une case de l'interface graphique en vert. La première balle du robot doit afficher le nombre de balles qui est trainé par le robot, alors que les autres balles sont représentées par un espace.

niveauEstComplet

Cette fonction publique reçoit :

- la grille contenant les obstacles du jeu.

La procédure parcourt toutes cases de la grille de jeu. Elle retourne "false" si la grille contient au moins un centre de distribution. Si la grille ne contient aucun centre, la fonction retourne "true".

viderGrille

Cette procédure publique reçoit :

- la grille contenant les obstacles du jeu.

La procédure écrase le contenu de toutes les cases de la grille reçu en y insérant une case vide.

restaurerGrilleDepart

Cette procédure publique reçoit :

- la grille contenant les obstacles du jeu.

La procédure écrase le contenu de toutes les cases de la grille reçue en y copiant le contenu des cases de la grille conservée lors du démarrage d'une nouvelle partie.

effectuerTour

Note : Une collision entre un obstacle et le robot s'étale sur plusieurs tours du jeu.

Cette procédure publique reçoit :

- l'interface graphique dans lequel le résultat du tour doit être affiché.
- la grille contenant les obstacles du jeu.
- le robot.
- les statistiques du jeu

Le déroulement d'un tour du jeu doit respecter la séquence suivante :

1. On vérifie si l'on doit continuer à gérer une collision détectée lors d'un tour précédent.
2. Lorsque c'est le cas, on ajuste la grille suite à cette collision.

3. Lorsqu'il n'y a pas eu de collision au tour précédent

4. Le robot est déplacé.
5. On vérifie si une collision est détectée.
6. Lorsque c'est le cas, on ajuste la grille suite à cette collision.

7. On déplace les obstacles dans la grille de jeu (collision ou non).

8. On effectue une pause de 200 ms.

Détection des collisions

Les collisions sont détectées lorsque le robot se situe dans une case en dessous d'un obstacle.

Ajustement suite à une collision

Lorsqu'une collision survient, il faut déterminer le type d'obstacle rencontré en fonction de la valeur placée dans la case au-dessus de la tête du robot.

Une valeur négative indique la rencontre d'un centre de distribution. Le robot perd une balle, ce qui incrémente la valeur du centre de distribution d'une unité et donne 2 points à l'utilisateur dans cette partie. Si c'est le cas, on met à jour le pointage maximal obtenu dans une partie.

Une valeur positive indique la rencontre d'un groupe de balles d'énergie. Le nombre de balles conservé par le robot est incrémenté d'une unité et on ajoute une balle dans la chaine de balle trainée par le robot (à noter : les coordonnées conservées sont la position de la nouvelle case dans la grille). La valeur du groupe de balles avec lequel la collision est survenue est décrémentée d'une unité. S'il devient nul, un nouveau groupe de balles est généré sur la première ligne de la grille dans une colonne VIDE choisit au hasard.

La case représentant la tête du robot est dessinée en noire pour mettre en évidence la collision.

Déplacement du robot

L'utilisateur déplace le robot à l'aide des flèches de gauche et de droite. Il est possible de connaître la touche sur laquelle l'utilisateur a appuyé en utilisant un des services publics de l'interface graphique. À noter que ce n'est pas le robot qui bouge dans la grille. Celui-ci restera toujours au centre de la grille.

Lorsque l'utilisateur appuie sur la flèche droite, toutes les balles actuellement contenues dans le robot sont décalées d'une case vers la droite. Une nouvelle balle, contenant la nouvelle position du robot, est ajoutée en tête du tableau. Puisque le nombre de balles n'est pas modifié, la dernière balle aura été "effacée".

Lorsque l'utilisateur appuie sur la flèche gauche, toutes les balles actuellement contenues dans le robot sont décalées d'une case vers la droite. Une nouvelle balle, contenant la nouvelle position du robot, est ajoutée en tête du tableau. Puisque le nombre de balles n'est pas modifié, la dernière balle aura été "effacée".

À chaque tour, l'on considère que le robot avance dans la grille. Toutes les balles actuellement contenues dans le robot sont décalées d'une case vers le haut. Une nouvelle balle, contenant la nouvelle position du robot, est ajoutée en tête du tableau. Puisque le nombre de balles n'est pas modifié, la dernière balle aura été "effacée".

Afin de garder le robot centrer, il faut incrémenter le numéro de ligne de la coordonnée de chaque balle du robot.

Déplacement des obstacles

Afin de simuler l'avancement du robot, à chaque tour, tous les obstacles doivent être décalés d'une ligne vers le bas du tableau. Afin de créer un niveau circulaire, la dernière ligne du tableau doit être recopiée dans la première ligne.

5. Contraintes de l'enseignant

Votre programme devra respecter les contraintes suivantes :

- Il devra respecter les normes de programmation présentées par votre enseignant. D'ailleurs, la qualité du code vaut au moins autant de points que le fonctionnement du programme. Le barème de correction peut différer pour chaque enseignant.

  Voici une liste non exhaustive des pratiques pénalisables :

  1. Identificateurs de fonction ou de procédure (nom + verbe) ne respectent pas la norme.
  2. Identificateurs non significatifs (variables, constantes, sous-programmes, …).
  3. Aération et/ou indentation et/ou impression laissent à désirer (80 colonnes max).
  4. Découpage en sous programmes insuffisants.
  5. Répétition inutile de code dû au manque de sous programmes.
  6. Répétition inutile de code dû à l'incompréhension de l'utilité du paramétrage.
  7. Constantes non définies.
  8. Constantes non utilisées lorsque possible (même dans les commentaires).
  9. Constantes en minuscules
  10. Commentaires non judicieux ou inutile.
  11. Commentaires des variables manquants.
  12. Commentaires des constantes manquants.
  13. Commentaires de spécification des sous-programmes manquants (quoi).
  14. Commentaire d'en-tête de programme manquant (explication, auteurs et version).
  15. Commentaires manquants sur la stratégie employée dans chaque sous-programme dont l'algorithme n'est pas évident et nécessite réflexion (comment).
  16. Commentaires mal disposés.
  17. Non-respect de l'encapsulation
  18. Non utilisation d'un sous-programme lorsque c'est possible ou exigé.
  19. Code inutile.
  20. Sous-programme qui fait plus d'une tâche.
  21. Affichage dans une fonction de calcul.
  22. Qualité du français dans les commentaires.
  23. Non-respect des droits d'auteurs.
  24. Non utilisation de boucle lorsque possible.

- Il ne devra contenir aucune variable globale. La présence d'une variable globale entraînera la perte de 25 % des points.
- Il ne devra contenir aucun `Goto` et aucun `Exit`. La présence d'une de ces instructions entraînera la perte de 10 % des points.

En cas de doute, consultez votre enseignant avant la remise!

Bon travail!