Compte rendu projet Informatique

Institut Supérieur d'Electronique de Paris

Oscar BOURAT
Guillaume CARRE

Table des matières

I. Le sujet

1. Cadre du projet

Le projet d'informatique du premier semestre de l'année 2015-2016 consiste en la création d'un jeu en langage de programmation Java. Le but de se projet est de nous familiariser avec le langage de programmation orienté objet. Nous devrons, pour mener à bien notre objectif, créer à la fois un jeu jouable en mode console, mais également en mode graphique. Nous allons ainsi devoir maîtriser un panel de fonctionnalités propres à la création de classes et d'objets, et également hiérarchiser notre code pour le rendre le plus intelligible et optimisé possible.

Le projet se poursuivra avec la création d'une intelligence artificielle à plusieurs niveaux. Cela nous permettra de ne pas uniquement s'arrêter à la création d'un plateau de jeu, mais également d'instaurer une dimension dynamique à notre programme, qui pourra ainsi être utilisable et proposera un challenge au joueur, même lorsqu'il joue seul.

2. Objectifs du projet

Le jeu que nous allons devoir créer s'intitule « *dots and boxes* ». Ce jeu ayant pour but de confronter deux joueurs, jouant tour par tour, sur un plateau de jeu délimité. Ce plateau de jeu comporte des points disposés uniformément, entre lesquels les joueurs, jouant chacun leur tour, traceront des traits reliant ces deux points. Lorsque qu'un joueur ferme un carré de quatre traits, il marque un point, et peut rejouer. Le joueur possédant le plus grand nombre de points lorsque tous les carrés sont fermés gagne la partie.

L'utilisateur devra pouvoir sélectionner un certains nombres d'options avant des commencer la partie, parmi lesquels :

- Choisir la taille du plateau, entre 2 et 5, correspondant au nombre de carrés par coté du plateau ;
- Choisir le handicap, correspondant au nombre de barres pré positionnées sur le plateau de jeu ;
- Choisir le mode de jeu, entre un joueur versus joueur et joueur versus IA. Dans le deuxième cas, le joueur pourra choisir le niveau de difficulté de l'IA parmi trois niveaux ;
- Entrer les noms des joueurs. Les joueurs seront différenciés par leur initiale pendant la partie.

Le mode graphique apportera des spécificités supplémentaires. En mode joueur versus joueur, les deux personnes possèderont une couleur spécifique permettant de les différencier lorsque ce que sera à leur tour de jouer. Lorsque l'on passera la souris entre deux points ou il est possible de jouer, une barre de la couleur du joueur apparaîtra en surbrillance. Lorsque le joueur joue à un emplacement, la barre jouée s'affichera en rouge.

II. Modélisation

1. Choix des classes

Le projet sera réalisé en créant plusieurs classes, conformément à la structure de programmation orientée objet.

Pour créer notre projet, nous avons utilisé un total de six classes, parmi lesquelles :

- Une classe « **Console** », destinée à contenir les attributs, méthodes et fonctions du jeu en mode console ;
- Une classe « **Graphique** », destinée à contenir les attributs, méthodes et fonctions du jeu en mode graphique ;
- Une classe « **Grid** », destinée à contenir les attributs, méthodes et fonctions permettant de créer et faire évoluer le plateau de jeu au cours de la partie ;
- Une classe « **Line** », destinée à contenir les attributs, méthodes et fonctions permettant de créer les lignes placées par les joueurs au cours de la partie ;
- Une classe « **Player** », permettant de gérer les spécificités de l'entité joueur pour que chaque joueur puisse être différencié pendant la partie ;
- Une classe « **Point** », permettant de calculer le nombre de point en fonction des éléments joués au cours de la partie par les différents joueurs.

2. Description des classes

a. Classe « Grid »

Il s'agit de la classe mère. Elle va gérer le fonctionnement du jeu, ainsi que le calcul des points. Nous avons commencé notre projet en créant cette classe. Cette classe dispose de plusieurs méthodes :

- **reset()** réinitialise les attributs de la classe quand le joueur souhaite rejouer ;
- **checkPartie()** vérifie si la partie est terminée, et retourne « true » si la partie est terminée, et « false » dans le cas contraire ;
- **generatePoints()** permet de générer les points en fonction des actions es joueurs ;
- **generateElements()** génère les lignes et les joueurs du plateau de jeu ;
- **aleatDashed()** permet de placer une ligne pointillée sur la grille de manière aléatoire. Cette méthode retourne « true » si la ligne est placée, et « false » sinon ;
- **generateDashed()** permet de générer un les lignes pointillées qui seront ensuite placée aléatoirement en appelant la méthode aleatDashed(). Cette méthode prend pour paramètre le nombre de lignes en pointillés ;
- **remainingLine()** donne le nombre de lignes restantes sur la grille. Cette méthode permet d'incrémenter une variable « count » qui fait office de compteur dans le programme ;
- **checkSquare(Player p)** permet de vérifier si un nouveau carré a été formé par le joueur p prit en paramètre. Elle retourne « true » si un nouveau carré est formé, et « false » sinon ;
- **setSquare(Player p, int c, int l)** est une méthode d'affichage permettant d'afficher l'initiale du joueur ayant formé un carré. Elle prend en paramètres le joueur p, le numéro de ligne l et le numéro de colonne c ;
- **lineType(int lpo, int rpo, String type)** permet de renvoyer le tracé de la ligne en fonction de sa nature, horizontal ou vertical et son type, sous forme de ligne pointillés, vide ou pleine. Cette méthode prend en paramètres le point de départ et d'arrivée de la ligne, ainsi que son type ;
- **setLine(int a, int b, String type)** modifie une ligne présente à une position particulière avec une nouvelle ligne de type différent. Elle prend en paramètres la colonne a et la ligne b et le type de la ligne. Grâce à cette méthode, le joueur peut placer une ligne sur le jeu pour tous les types de lignes ou il est possible de jouer. Elle retourne « true » si la ligne est modifiée, et « false » sinon ;

- **getWinner (Player p1, Player p2)** prend en paramètre les deux joueurs et compare leur score. Elle renvoie le nom du joueur ayant gagné et en cas d'égalité, renvoie « null » ;
- **main(String[] args)** est la méthode principale du programme. Elle propose à l'utilisateur de choisir entre le mode console et le mode graphique. En fonction de la réponse du joueur, elle appelle la méthode principale du mode graphique ou du mode console.

En créeant la classe « Grid », nous avons également crée simultanément les classes « Line », « Player » et « Point », car celle-ci fonctionne en utilisant des méthodes de ces classes. Analysons à présent ces différentes classes.

b. Classe « Line »

Cette classe permet de gérer toutes les manipulations concernant les lignes du plateau de jeu. Elle contient des méthodes permettant de vérifier l'état des lignes, mais aussi de les dessiner ou encore de les modifier. Ces méthodes sont les suivantes :

- **isHorizontal()** vérifie si la ligne est horizontale ou non. Elle renvoie « true » si la ligne est horizontale, et « false » sinon ;
- **isTaken()** vérifie si la ligne est prise. Elle renvoie « true » si c'est le cas et « false » sinon ;
- **draw()** permet de dessiner graphiquement la ligne. On utilise ici la classe StdDraw, nous permettant de tracer l'entité ligne et de lui associer une couleur. Rouge pour une ligne tracée et grise pour une ligne pointillée.
- **hitboxLine()** permet de prendre en paramètre une ligne, définie par les coordonnées de la souris au moment ou le joueur clique sur la grille, et de modifier son état. La fin du tour du joueur est ensuite déclenchée si la ligne jouée est valide (à l'état « dashed » ou « solid »).

c. Classe « Player »

Cette classe permet de gérer tout ce qui concerne l'objet ligne. Les méthodes permettent de gérer l'ajout de lignes au tableau de jeu, mais également l'interaction des joueurs et de l'IA avec les lignes. Ces méthodes sont les suivantes :

- **move()** demande au joueur les coordonnées d'une ligne et vérifie l'entrée, puis regarde si cette ligne est disponible ;
- **draw()** affiche graphiquement l'initiale du joueur sur la grille. Permettant d'identifier quel carré à été rempli par tel joueur ;
- **playRandom()** permet de jouer aléatoirement entre deux points disponibles si aucune ligne n'est déjà tracée, par une méthode récursive. Cette méthode renvoie la ligne générée, ou null si elle n'existe pas. Notre IA de premier niveau utilise cette méthode. Cela revient à jouer au hasard ;
- **setLevel()** est une méthode simple permettant au joueur de sélectionner le niveau d'IA choisis via un menu. En fonction du choix du joueur, elle fait appel aux méthodes correspondant aux IA ;
- **level0(Object[][] grid)** correspond au premier niveau d'intelligence artificielle. Celle-ci n'est pas bien évoluée et fait appel à la méthode jouerRandom(), ce qui revient à jouer aléatoirement
- **level1(Object[][] grid)** correspond au deuxième niveau d'intelligence artificielle. Elle est plus évoluée que le premier niveau car elle ferme les carrés ayant 3 côtés fermés. Cela permet d'instaurer plus d'adversité. L'IA joue au hasard si le cas précédemment cité ne se présente pas ;
- **isLineExisting(List<Line> list, line l)** permet de vérifier si la ligne est présente à un endroit donnée. Cette méthode va nous servir pour programmer l'intelligence artificielle de troisième niveau.
- **level2(Object[][] grid)** correspond au troisième niveau d'intelligence artificielle. Elle possède le niveau d'avancement le plus évolué, en fermant les carrés ayant 3 côtés fermés, et en évitant de rajouter un troisième coté aux carrés

possédant déjà deux cotés fermés. L'IA joue au hasard si les cas précédemment cités ne se présentent pas.

d. Classe « Point »

Cette classe gère tous les éléments les points formant la grille de jeu. La méthode draw() de cette classe permet d'afficher graphiquement ces points.

e. Classe « Console »

Cette classe servira de classe d'instanciation pour le mode console du jeu. Les méthodes présentes sont les suivantes :

- **reset()** permet de réinitialiser les attributs de la classe lorsque qu'une personne souhaite rejouer ;
- **init()** initialise le jeu en affichant le menu, en prenant en paramètre les options choisies par l'utilisateur et lance la partie ;
- **isNumber()** permet de vérifier si le caractère rentré est bien un nombre. Dans le cas contraire, la méthode renvoie une erreur. Cette méthode va nous servir pour vérifier les entrées utilisateurs dans la sélection des menus.
- **drawSeparator()** trace une ligne de séparation de la longueur de la grille ;
- **menu()** lance le menu permettant de sélectionner la taille de la grille, et le mode de jeu ainsi que leurs options associées ;
- **Player[] setPlayingplayer()** demande le nom des joueurs et va créer un nouveau joueur possédant ce nom ;
- **turn(Player p)** permet de faire jouer un joueur ;
- **game(Player p1, Player p2)** permet de gérer l'alternance de jeu entre les deux joueurs ;
- **draw(Object[][] grid)** permet de dessiner la grille dans la console.

f. Classe « Graphique »

Cette classe servira de classe d'instanciation pour le mode graphique du jeu. Les méthodes présentes sont les suivantes :

- **init()** initialise le jeu en affichant l'interface graphique, le menu, en prenant en paramètre les options choisies par l'utilisateur et lance la partie ;
- **reset()** permet de réinitialiser les attributs de la classe lorsque qu'une personne souhaite rejouer ;
- **isKeyReleased(int key)** permet de vérifier si la touche à été relâchée et donc que le joueur à bien tappé à cet endroit. Cette méthode renvoie true lorsque la touche est relâchée et false si la touche n'est pas appuyée ;
- **mouseReleased()** teste si le bouton de la souris à été relâchée et donc que le joueur à bien cliqué à cet endroit. Cette méthode renvoie true lorsque le bouton est relâchée et false si le bouton n'est pas appuyée ;
- **isMouseInArea(double x0, double y0, double x1, double y1, double ; epaisseur)** détecte si la souris se situe dans une aire ;
- **drawTextArea()** à pour fonction d'afficher une zone de texte modifiable. Cela va nous permettre d'afficher des informations sur l'interface de jeu que l'on pourra modifier ensuite ;
- **drawLabel(String content,Color background,Color text,double x,double y,Font font,double width,double height)** affiche du texte avec un fond de couleur ;
- **drawButton()** permet de créer un objet bouton cliquable ;
- **isButtonPressed(double x, double y, double width, double height)** permet de détecter l'activation d'un bouton si la souris est placée sur une certaine position. Cette méthode permet ainsi de gérer la hitbox des boutons ;

- **drawStepper(double x, double y, int start, int end, int step, int compteur)** crée un compteur permettant de compter le nombre de fois où les boutons sont pressés pour ainsi faire varier les options de difficulté ou de handicap dans le menu.
- **launchMenu()** lance le menu permettant de sélectionner le mode de jeu, la taille de la grille, la difficulté de l'IA, le nombre de traits en pointillés (handicaps) et le nom du ou des joueurs ;
- **detectPos(double x0, double y0, double x1, double y1, double epaisseur)** permet de détecter si la souris se trouve dans une aire où un élément peut être joué.
- **hover(Player currentplayer)** détecte si la souris passe entre deux points pouvant être reliés. Si c'est le cas, un trait en surbrillance de la couleur du joueur s'affiche ;
- **showHeader(Player p, int size, Player gagnant)** permet d'afficher le texte au dessus de la grille de jeu. Ce texte affiche le tour du joueur ainsi que le résultat final de la partie ;
- **showScore(Player p1, Player p2)** permet d'afficher le texte présent sous la grille de jeu. Il s'agit du score des joueurs représentés par leurs initiales et leurs couleurs respectives ;
- **setLine(Object[][] grid)** permet de modifier une ligne dans la grille de jeu ;
- **draw(Object[][] grid)** trace le plateau de jeu ;
- **displayUpdate(Player p, Player gagnant)** met à jour l'affichage de la grille de jeu en fonction des mouvements joués par l'utilisateur ainsi que l'affichage au dessus de la grille, indiquant le tour du joueur et les message de fin de partie, et enfin l'affichage en dessous de la grille en faisant évoluer le score des participants ;
- **turn(Player p)** fait jouer le joueur, et rejouer dans le cas ou il ferme un carré ;
- **game(Player p1, Player p2)** gère le jeu et permet de relancer celui-ci lorsque la partie est terminée.

III. Fonctionnement du jeu

1. Démarrage du jeu

Pour démarrer le jeu, on éxecute la classe Grid. La console nous donne ensuite le choix de jouer dans la console ou en mode graphique.

2. Mode console

Avant de commencer à jouer, on demande au(x) joueur(s) de renseigner certains paramètres du jeu :

- Taille de la grille (2-3-4-5). *Nombre de carré par ligne ou colonne* ;
- Mode de jeu (1-2). *Le mode de jeu 1 fait jouer deux joueurs. Le mode de jeu 2 fait jouer un joueur contre un joueur artificiel* ;
- Nombre de handicaps (0-…). *Nombre de trait en pointillé placé aléatoirement sur la grille* ;
- Mode de jeu 2 **UNIQUEMENT**: Niveau de l'IA(1-2-3) ;
- Joueur commençant à jouer en premier -*1 pour premier joueur, 2 pour 2e joueur*. Nom du/des joueur(s).

Ensuite le jeu commence par le premier joueur renseigné. Le joueur joue lorsque son nom s'affiche sur la console et l'invite à jouer.

Pour jouer, le joueur doit renseigner **deux points**. L'ordre des points n'a pas d'importance du moment que l'entrée est valide et qu'il n'y a pas de trait déjà présent à l'emplacement spécifié. Pour renseigner ces deux points, le joueur doit les taper dans la console séparée par une virgule : 1,2 avant d'appuyer sur ↩ pour valider.

Lorsque le jeu est terminé, le(s) joueur(s) a/ont la possibilité de rejouer en tapant le nombre 1.

3. Mode graphique

Avant de commencer à jouer, on demande au(x) joueur(s) de renseigner certains paramètres du jeu :

- Mode : *Appuyez sur les boutons pour sélectionner le mode de jeu* :
  - J1 vs J2 - *Deux joueurs jouent chacun leur tour ;*
  - J1 vs Ordi - *Un joueur joue contre une IA* ;
- Taille : (*idem mode console*)
  - Appuyez sur le bouton + ou - pour augmenter ou diminuer la taille ;
- Handicap (*idem mode console*)
  - Fonctionnement identique à **Taille** ;
- **J1 vs Ordi UNIQUEMENT** : Difficulté -*Niveau de l'IA*
  - Fonctionnement identique à **Taille** ;
- NomJ1/NomJ2 :
  - Cliquez sur la zone de texte du nom du joueur pour la modifier ;
  - Appuyez sur la touche retour pour supprimer des lettres ;
- 1er Joueur - *Appuyez sur les boutons pour sélectionner le joueur démarrant la partie*
  - J1 - *J1 commence à jouer ;*
  - J2 ou Ordi - *J2 ou Ordi commence à jouer.*

Ensuite le jeu commence par le 1er joueur renseigné. Pour créer des lignes ou en effacer (traits en gris), il suffit de passer la souris entre deux points, une ligne de la couleur du joueur apparait alors et le joueur n'a qu'a cliquer pour créer la ligne. C'est ensuite au tour du joueur suivant.

Lorsque la partie est terminé, le(s) joueur(s) peut/peuvent recommencer une nouvelle partie en cliquant sur le bouton central.

IV. Description des fonctionnalités spécifiques

1. IA

a. Niveau 1

- Choisi un emplacement aléatoire.

b. Niveau 2

- Cherche à fermer un carré dont il reste un emplacement vide ;
- Choisi un emplacement aléatoirement sinon.

c. Niveau 3

- Evite de choisir les emplacements d'un carré possédant déjà deux côtés ou possédant 3 côtés dont un en pointillé ;
- Cherche à fermer un carré dont il reste un emplacement vide sinon ;
- Choisi un emplacement aléatoirement sinon.

2. Mode console

a. Vérification des champs texte

- Sélection des paramètres - *Vérification des valeurs entrées* :
  - Sont des Nombres ;
  - Sont parmi les valeurs autorisées
    - Cas des handicap - *valeurs autorisées comprises entre 0 et le nombre de trait en pointillé que peut contenir une grille de la taille sélectionnée*.
- Coordonnés des lignes - *Vérifications* :
  - Si les coordonnés sont entrés sous la bonne forme ;
  - Si les coordonnés correspondent à une ligne valide ;
  - Si les coordonnés correspondent à une ligne non existante.

Remarque

On peut entrer aucune valeur pour certains champs lors de la sélection des paramètres du jeu. Des valeurs par défaut sont alors sélectionnées (les valeurs les plus faibles des paramètres).

3. Mode graphique

a. Hover

Lorsqu'un joueur passe la souris à l'emplacement d'un trait et qu'il peut modifier l'emplacement, un trait de la couleur du joueur (couleur du nom en haut de la fenêtre) s'affiche.

Lorsque l'on passe la souris sur un bouton et qu'il change de couleur, cela signifie que l'on peut cliquer dessus pour effectuer une action.

b. Sélection des valeurs numériques

Lorsque l'on choisi le nombre de traits en pointillé, le sélecteur se bloque automatiquement au nombre de trait maximum pour la taille de la grille sélectionnée. Si l'on diminue la taille de la grille et que le nombre de traits choisi initialement était supérieur au nombre maximal de traits pour la taille actuelle, le nombre de traits en pointillé se règle automatiquement sur le nombre de traits maximum de la grille actuelle.