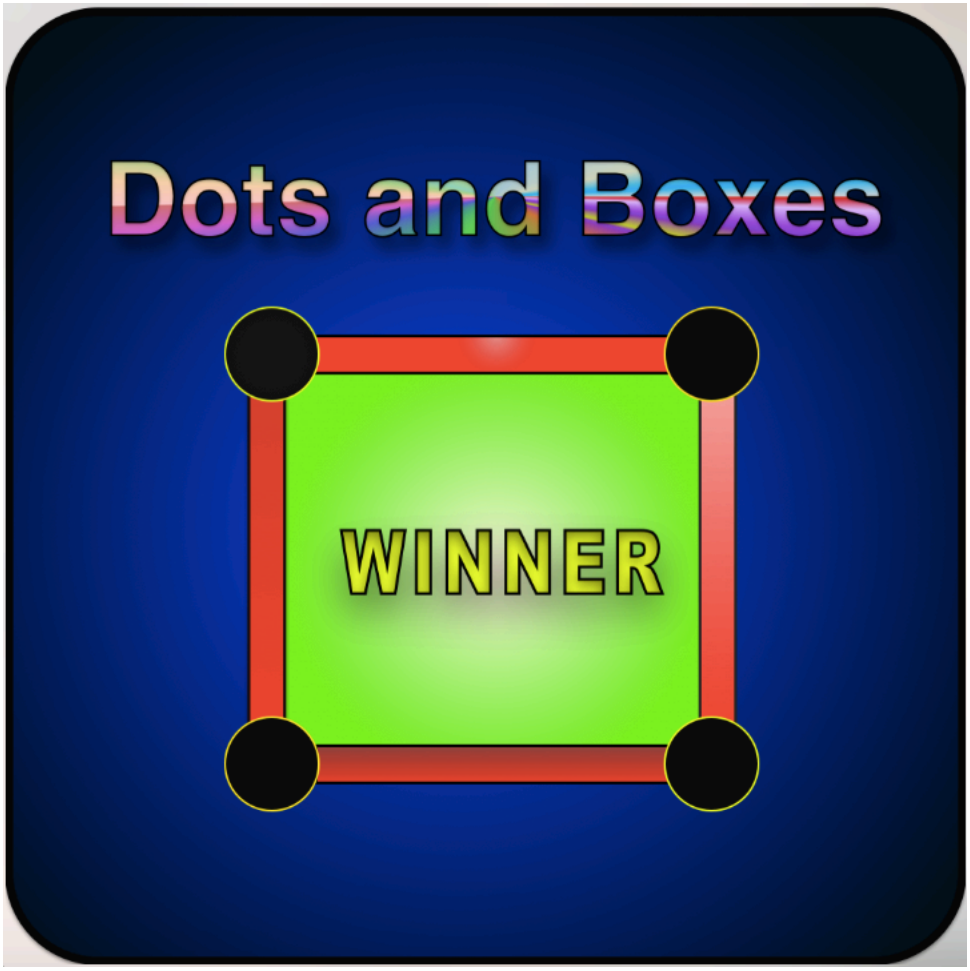
Projet Informatique



Binôme :

- Arnaud Ribeyrolles ;
- Jean-Baptiste Maigrot.

Sommaire

I. Introduction

Le projet d'informatique 2015/2016 consiste en la création d'un jeu dans le langage de programmation Java.
Le jeu à programmer est « Dots and Boxes », la « Pipopipette » en français. C'est un jeu de société se pratiquant à deux personnes, tour à tour.
Le but est de tracer des traits autour de carrés, afin de former ceux-ci.

Celui qui réussit à fermer un carré marque un point. Celui qui a le plus de point à la fin de la partie gagne.

I. Rappel des Consignes

1°) Déroulement du déroulement du jeu attendu :

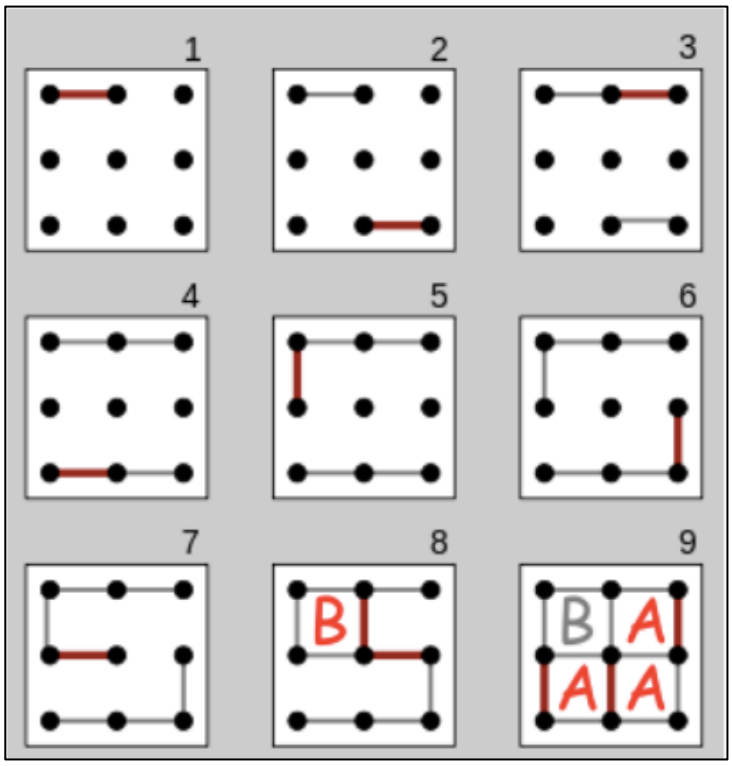
Choix au lancement :

- Graphique / console
- 2 joueurs / 1 joueur contre l'IA
- Taille de la grille
- Nombre de lignes pointillées

Afficher la grille (à jour) à chaque tour.

Lorsque la partie est terminée, demander si l'on veut rejouer.

2°) Exemple de l'aspect d'une partie en mode Graphique attendue :



II. Classes

- **JeuCarre :** contient l'algorithme de démarrage et choix du mode de jeu,
- **Plateau :** contient l'état du plateau et les fonctions liées (affichage et modification),
- **Menu :** contient les fonctions liées aux menu en mode graphique,
- **Graphique :** contient les fonctions liées à l'affichage du jeu en mode graphique,
- **CalculsIA :** contient les fonctions de l'intelligence artificielle.

III. Algorithme

1°) Déroulement général de notre jeu.

Initialisation des variables

Mode console ou graphique ?
Si Console → Demande les paramètres du jeu en console.
Si Graphique → Ouvre les menus graphiques.

Initialisation du plateau
Si Console → Fonctions Initialisation() & Pointilles().
Si Graphique → Fonctions Initialisation(), Pointilles() & IntialisationGraphique().

Jeu
Tant que la grille n'est pas remplie :
Tour de jeu
Choix du trait par le joueur (ou calcul par l'IA)
Fonction Update : Place le trait (ou enlève le trait pointillé)
Fonction Cocher : - Remplit la(les) case(s) complétée(s)
- Incrémente le compteur du joueur
Si le joueur n'a pas complété de case :
Changement de joueur

Fin de la partie. Rejouer ?
Oui → Retour au début du programme
Non → Arrêt du programme

2°) Méthodes les plus importantes détaillée précisément.

Fonction boolean **Update** (ligne, colonne, joueur, plateau) :
    Si ligne paire et colonne impaire :
        Vide → Trait horizontal
        Pointillé → Vide
    Si ligne impaire et colonne paire :
        Vide → Trait vertical
        Pointillé → Vide
    Fonction Cocher
    Renvoyer *rejoue*

NB : Si le joueur demande de jouer à la ligne 2 et colonne 3 en console, celle-ci se trouve en position [1,2] dans le tableau représentatif du plateau ! Ceci est pour une raison esthétique : le joueur n'aura pas à jouer en position 0/1 par exemple, mais en 1/2.

Fonction boolean **Cocher** (ligne, colonne, joueur, plateau) :
    Par défaut, la variable *rejoue* = false
    Si trait horizontal :
        Si l'on complète une case au-dessus ou en-dessous :
            Mettre le numéro du joueur dans la(les) case(s) complétée(s)
            Mettre à jour le score
            rejoue = true
    Si trait vertical :
        Si l'on complète une case à gauche ou à droite :
            Mettre le numéro du joueur dans la(les) case(s) complétée(s)
            Mettre à jour le score
            rejoue = true
    Renvoyer *rejoue*

Fonction int[] **CalculsIA1** (plateau) :
    CasesVides = CréationCasesVides(plateau) → Toutes les cases disponibles
    hauteuraléatoire= nombre aléatoire entre 0 et longueur(CasesVides)
    tab[0] = ligne aléatoire comprise dans CasesVides à hauteuraléatoire.
    tab[1] = colonne aléatoire comprise dans CasesVides à hauteuraléatoire.
    Renvoyer tab[LigneChoisie, ColonneChoisie]

Fonction int[] **CalculsIA2** (plateau) :
    CasesVides = CréationCasesVides(plateau)
    Balayage du plateau (par case) :
        Si une case peut être complétée (3 traits sont déjà placés) :
            Renvoyer coordonnées du trait correspondant dans tab[]
    Si aucune case ne peut être complétée :
        hauteuraléatoire= nombre aléatoire entre 0 et longueur(CasesVides)
        tab[0] = ligne aléatoire comprise dans CasesVides à hauteuraléatoire.
        tab[1] = colonne aléatoire comprise dans CasesVides à hauteuraléatoire.
    Renvoyer tab[LigneChoisie, ColonneChoisie].

Fonction int[] **CalculsIA3** (plateau) :

CasesVides = CréationCasesVides(plateau)

Balayage du plateau (par case) :

Si une case peut être complétée (3 traits sont déjà placés) :

Renvoyer le trait correspondant

Si aucune case ne peut être complétée :

CoupsInterdits : liste des coups qui permettraient à l'autre de compléter une case au tour suivant.

CoupsJouables = CasesVides - CoupsInterdits

Si CoupsJouables n'est pas vide :

hauteuraléatoire= nombre aléatoire entre 0 et longueur(CoupsJouables)

tab[0] = ligne aléatoire comprise dans CoupsJouables à hauteuraléatoire.

tab[1] = colonne aléatoire comprise dans CoupsJouables à hauteuraléatoire.

Si CoupsJouables est vide :

hauteuraléatoire= nombre aléatoire entre 0 et longueur(CasesVides)

tab[0] = ligne aléatoire comprise dans CasesVides à hauteuraléatoire.

tab[1] = colonne aléatoire comprise dans CasesVides à hauteuraléatoire.

Fonction boolean **Hitbox**(int joueur, String[][] visuel) :

Largeur du rectangle de détection pour les clicks

Si Joueur clique sur un endroit vide ou sur un Pointillé, Fonction detectPos()

Update et Rejoue si case complétée.

Si Joueur clique sur un endroit déjà joué, ne le fais pas rejouer.

Fonction boolean **detectPos**(int ligne, int colonne, int joueur, String[][]visuel, double x0, double x1, double y0, double y1) :

double x : Position horizontale de la souris.

double y : Position verticale de la souris.

Si (coordonnées souris) ∈ (coordonnées x0, y0, x1, y1) rectangle de détection :

Update le trait de coordonnées ligne/colonne où le joueur a cliqué.

3°) Méthodes secondaires décrites plus brièvement.

CLASSE JeuCarre

Fonction main() :

→ Appelle la fonction Jeu()

Fonction Jeu() :

→ Demande le mode de jeu

→ Distribue les taches de demande d'informations

→ Initialise le tableau du plateau de jeu

→ Distribue les taches de déroulement des modes de jeu.

Fonction DemandeInformation() :

→ Demande les informations en console

Fonction Demandegraphique() :

→ Fais appel à la **CLASSE Menu**

Fonction JoueurVsJoueur() :
→ Différent si mode graphique ou console.
→ Tant que *demande* est vrai, le même joueur continue du jouer.
→ Si le *compteur* est égal au nombre de cases totales, break while(demande).

Fonction JoueurVsOrdi () :
→ Différent si mode graphique ou console.
→ Différent si tour de l'ordi ou tour du joueur.
→ Tant que *demande* est vrai, l'ordi/joueur continue du jouer.
→ Si le *compteur* est égal au nombre de cases totales, break while(demande).

Fonction OrdiVsOrdi() :
→ Différent si mode graphique ou console.
→ Tant que *demande* est vrai, le même ordi continue du jouer.
→ Si le *compteur* est égal au nombre de cases totales, break while(demande).

Fonction changer() :
→ Si (joueur==1) => renvoyer (joueur=2)
→ Si (joueur==2) => renvoyer (joueur=1)

Fonction TestSiRange() :
→ Test si le nombre rentré au préalable est dans une fourchette précise.

Fonction TestSiNombre() :
→ Test si le nombre rentré au préalable est un nombre et pas un caractère.

Fonction Reset() :
→ Evite les problèmes de variables statiques lorsqu'on recommence le jeu.

CLASSE Menu

Fonction Menu1() :
→ Génère le premier menu où les demandes d'informations se font.

Fonction Menu2() :
→ Génère le deuxième menu où la suite demandes d'informations se font :
→ Noms des joueurs
→ Modification si nécessaire
→ Lancement du jeu.

Fonction DetectPos() :
→ Créer un rectangle qui sera utile pour la hitbox du click souris.

Fonction Bouton() :
→ Permet de créer un bouton d'une taille précise, avec un texte précis à l'intérieur centré, et qui change de couleur quand on passe la souris dessus.

Fonction BoutonEntreText() :
→ Même fonction que précédement, mais permet de positionner le texte à l'intérieur du rectangle comme on le souhaite.

Fonction Cover() :

→ Permet de faire des effets de changement de couleur au passage de la souris dans la hitbox du bouton.

Fonction CoverText() :

→ Même fonction que précédement, mais permet de positionner le texte à l'intérieur du rectangle comme on le souhaite.

Fonction TextEncadre() :

→ Permet de faire un bouton sans surligner quand la souris passe dessus.

Fonction isKeyReleased() :

→ Fonction qui permet d'éviter d'effacer tout le nom du joueur si la personne qui rentre les noms laisse appuyer la touche d'effacement

→ Bloque la boucle tant que la touche est appuyée.

CLASSE Graphique

Fonction AfficherVisuelGraphique() :

→ S'occuper d'afficher en graphique le tableau du plateau de jeu (à jour).

→ Fait appel à AfficherTextGraphique()

Fonction AfficherTextGraphique() :

→ S'occuper d'afficher en graphique tous les textes.

Fonction detectPos() :

→ Permet d'Update le tableau du plateau de jeu là où le joueur a cliqué.

Fonction DetectPos() :

→ Créer un rectangle qui sera utile pour la hitbox du click souris.

CLASSE Plateau

Fonction TourJoueur() :

→ Gère le tour de chaque joueur.

→ Prend les entrées du joueur

→ Vérifie les erreurs d'entrées

→ Update si pas d'erreur d'entrée.

Fonction TourIA() :

→ Gère le tour de l'IA en mode Joueur vs IA & IA vs IA.

→ Selon la difficulté, renvoie à la bonne fonction dans **CLASSE CalculsIA**

→ Vérifie les erreurs d'entrées

→ Update si pas d'erreur d'entrée.

Fonction Verifier() :

→ Vérifie que le joueur a donné des coordonnées correctes.

Fonction UpdateOrdi() :

→ Même Fonction que Update décrite plus haut, mais change les textes en console.

Fonction CocherOrdi() :

→ Même Fonction que Cocher décrite plus haut, mais change les textes en console.

Fonction Afficher() :

→ Affiche le tableau du plateau de jeu (en mode console).

Fonction Initialisation() :

→ Crée le tableau du plateau de jeu.

Fonction Pointilles() :

→ Place les pointillés sur le plateau.

CLASSE CalculsIA

Fonction CreationCasesVides() :

→ Renvoie la liste des traits vides du tableau du plateau de jeu.

Fonction ClearSimilarites() :

→ Renvoie une liste de laquelle on a retiré les éléments d'une autre.

Fonction ClearTableau() :

→ Efface les zéros et les doublons

→ Renvoie une liste de la bonne forme.

NB : Pour plus de détails concernant ces fonctions secondaires, vous avez la possibilité de regarder le Javadoc fourni dans le fichier doc complémentaire (ouvrir index.html). Et pour encore plus de précision, les fonctions sont commentées dans le programme.

Conclusion :

Certains bonus ont été effectués par rapport au cahier des charges :

→ La possibilité d'atteindre une taille de plateau de jeu supérieur à 5, allant jusqu'à 49 sans avoir de décalage dans l'affichage.

→ La présence d'un mode IA vs IA qui fut très utile quand il fallait débugger des erreurs de conditions entre autres et plutôt distrayant à regarder.

→ La possibilité d'entrer des noms de joueurs.

→ Des tailles adaptatives de fenêtre selon la taille de plateau demandée par le joueur.

→ Des textes et tailles de points adaptatifs selon la taille de plateau demandée par le joueur.

→ Non pas un mais trois niveaux d'IA plus ou moins intelligents !