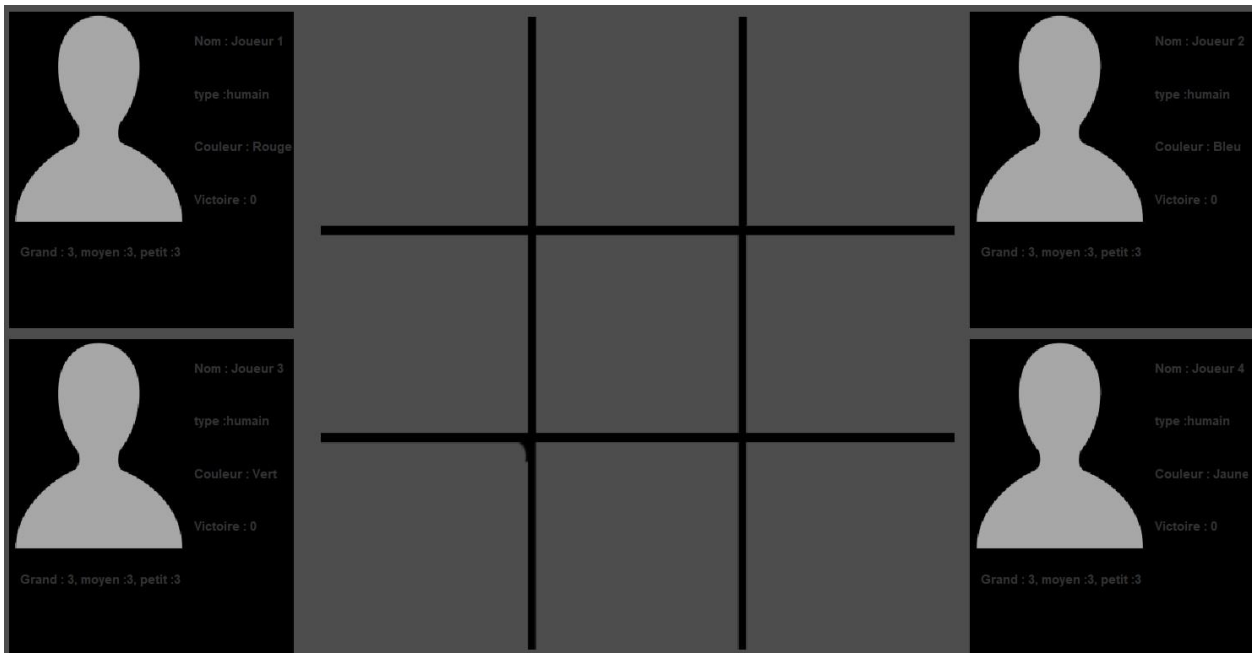


Projet Informatique S1

2022

—

FERON Maël
RAMDANI Noé
RAMAROSANDRATANA Steven
STH1-B5

https://www.canva.com/design/DAFW55IudlA/f59tyjKi_H-OFiKbcgvcmw/edit?utm_content=DAFW55IudlA&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton

Introduction

Ce dossier à pour but d'expliquer le déroulement de notre développement du programme "OTRIO" qui consiste en un morpion amélioré. Le programme est écrit en langage JAVAscript.

Pour mener à bien le projet il à tout d'abord fallu tenir compte des grands écarts de niveau dans le groupe pour ne laisser personne de côté. Il nous à ensuite fallu prendre connaissance du jeu, de ses règles et de ses subtilités afin de rédiger un algorithme puis mettre en place le code et le développer au fur et à mesure.

Objectifs de départ

1. Obtenir un jeu fonctionnel
2. Ajouter des IA de différents niveaux
3. Mettre en place une interface interactive

Compétences du groupe:

Nous avons directement remarqué que le groupe était très hétérogène en termes d'aisance vis à vis de la programmation. Cela ne facilite pas la tâche mais il faut s'adapter afin que chacun progresse et s'investisse. De plus, aucun de nous ne connaissait le langage JAVA avant le début d'année, il a donc fallu apprendre et faire des recherches.

STEVEN est particulièrement à l'aise et amateur de programmation / NOÉ n'est pas novice en programmation / MAËL a quelques bases mais a très peu d'expérience autre que les programmes faits en cours d'informatique.

Répartition des tâches:

1. Brainstorming collectif (quelles sont les idées de chacun)
2. Maël: choix des modes

   Steven: Déroulement du jeu

   Noé: Fonctionnement des IA
3. Ensuite Steven s'est occupé de créer une base solide du programme, Noé a continué de développer les modes de jeu et une IA plus intelligente et Maël s'est occupé de désigner les éléments du jeu et de les placer dans l'interface en les programmant.

Présentation de l'Algorithme

I. Idée de base

Premièrement, notre programme est centré sur une boucle dite infinie, avec un "while true" puis une suite de IF qui constituent les déplacements effectués dans les différents lieux. Pour chaque IF, il existe une combinaison unique de tests qui permet de changer le "décor" et donc d'amener dans un nouveau lieu. Sachant que l'interface graphique est très interactive avec l'utilisateur, il a été très amusant de programmer avec pour récupérer des informations sans faire cesser le déroulement interne du programme, grâce à les OVERRIDES, des interactions machine inter-humain.

II. Création de l'algo

Tout d'abord, notre algorithme de test Principale (de Victoire) est très optimisé, elle se base sur ces 2 matrices "clé", qui sont une matrice sélection et une matrice test.

[0][0]	[0][1]	[0][2]
[1][0]	[1][1]	[1][2]
[2][0]	[2][1]	[2][2]
[0][0]	[1][1]	[2][2]
[2][0]	[1][1]	[0][2]
[0][0]	[1][0]	[2][0]
[0][1]	[1][1]	[2][1]
[0][2]	[1][2]	[2][2]

[0]	[0]	[0]
[1]	[1]	[1]
[2]	[2]	[2]
[0]	[1]	[2]
[2]	[1]	[0]

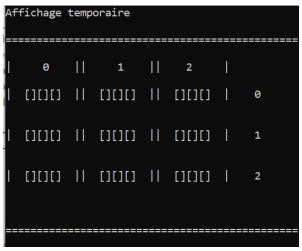
Grâce à ces matrices, nous allégeons l'écriture de notre programme évitant d'écrire prêt de 49 tests IF. Pour ce faire, nous avons 3 boucles qui traversent d'abord les couleurs, puis la matrice sélection et enfin la matrice test. Elle va à chaque tour de boucles changer de couleur, prendre 3 sets de Coordonnées (de case), puis une possibilité de gagner dans la matrice Test(Croissant / même taille / décroissant). Elle prend alors une ligne, effectue tous les tests de victoire possibles et si la Couleur dans la première boucle est retrouvée dans les cases avec le test sélectionné un drapeau est mis en true et break toutes les boucles. Néanmoins nous avons volontairement décidé de laisser dans une autre boucles plus simples les tests dans une même case car elle alourdit la matrice sélectionnée et effectue des tours supplémentaires inutiles dans la boucle.

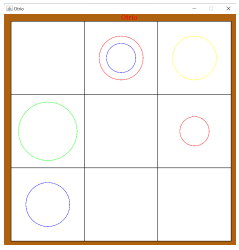
Voici ci-dessous l'algorithme de la fonction de test principal.

```
Fonction Test Principale (var t1 TAB 3*3*3 ):boolean
        Variables
        i, i1, i2 : entiers
        a :booléen
        Début
        a <- true;
        Pour i de 0 à couleurR.length faire    // parcourt les couleurs
        Pour i2 de 0 à Matselec.length faire  // parcourt la matrice Sélection
                Pour i3 de 0 MatTest.length faire   // parcourt la matrice Test
                Si (couleurR[i]==t1[Matselec[i2,0,0]][Matselec[i2,0,1]][MatTest[i3,0]] et couleurR[i]==t1[Matselec[i2,1,0]][Matselec[i2,1,1]][MatTest[i3,1]])
                et  couleurR[i]==t1[Matselec[i2,2,0]][Matselec[i2,2,1]][MatTest[i3,2]]))Alors
                 a= false
                  break
                fin Si
                fin Pour
                        si !a alors
                        break
                        fin SI
        fin Pour
                si !a alors
                break
                fin Si
                Si TestSecondaire(CouleurR[i]) Alors
                a<-true
                Fin Si
        fin Pour
                Si TestSecondaire
        TestPrincipale <-- a
        Fin
```

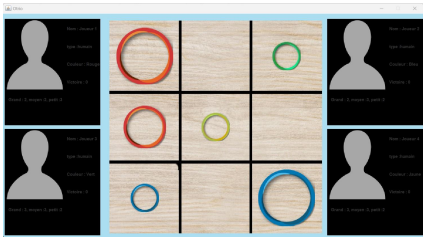
III. Création de l'interface

L'interface n'était pas un élément obligatoire mais ayant bien avancé sur le programme nous avons décidé de l'améliorer au fur et à mesure.





Voici à quoi ressemblaient les premiers affichages. Le premier s'affichait dans cmd, le deuxième utilisait l'extension canvas.



L'interface actuelle utilise swing et awt qui ouvrent plus de possibilités dans le design et les fonctionnalités. La première étape a été de créer un JFrame qui permet l'interaction avec le code sur lequel nous avons ajouté des couches JLayerPane permettant d'ajouter toutes sortes d'objets: image, bouton, texte ...

Il a ensuite fallu tout agencer grâce à des fonctions telles que "Nom.setBackground(Color=red) et rendre fonctionnel en ajoutant les boutons a la couche Interface.

IV. Développement des IA

Fonctionnement de l'IA noob :

La programmation de l'IA (noob) a été plutôt rapide. Pour cela, nous avons créé 3 variables (k1, k2 et k3) et donné une valeur aléatoire entre 0 et 2 à chacune. Chaque variable correspond à une donnée du coup qui va être jouée par l'IA : k1 la ligne, k2 la colonne et k3 la taille de l'anneau à jouer.

Deux vérifications sont ensuite effectuées par le programme :

-vérifier que la case est libre -vérifier que l'IA possède encore un anneau de cette taille

```
while (t1[k1][k2][k3]!=null || NombreDeCercle[i+2*CouleurChoisi[i]][k3]==0 ){
k1 = (int)(Math.random()*(max-min+1)+min);
k2 = (int)(Math.random()*(max-min+1)+min);
k3 = (int)(Math.random()*(max-min+1)+min);
}
```

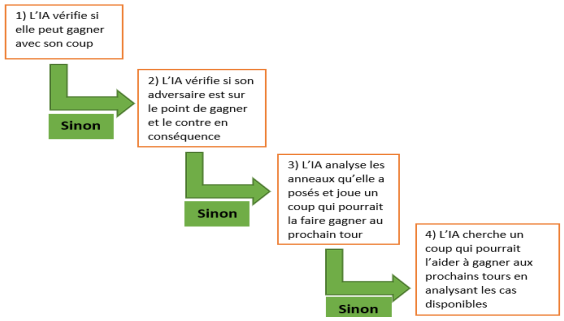
Tant que la première proposition OU la seconde n'est pas validée, le programme régénère trois valeurs à affecter aux variables k1, k2 et k3.

Fonctionnement de l'IA (expert) :

La programmation de cet IA a été beaucoup plus challengeante. Dans un premier temps, il nous a fallu réfléchir à comment l'IA allait réfléchir. Nous avons alors créé un arbre de priorités c'est-à-dire un ordre des tests à faire : si le premier test n'est pas validé, l'IA passe au second etc… Le premier test réalisé par l'IA est donc la vérification de la possibilité de gagner. Elle utilise alors 2 boucles imbriquées. Cette vérification repose sur les matrices "MatSelec" et "MatTest" présentées précédemment. Pour expliquer simplement le processus, nous rappelons que la "MatSelec" permet la sélection de groupes de cases (ligne, colonne, diagonale) et la MatTest permet d'effectuer les différents tests (anneaux de même taille alignés, anneaux alignés de façon croissante) etc… Ainsi, l'IA effectue l'ensemble des tests pour la première ligne, puis pour la seconde etc… jusqu'à avoir fait tous les tests pour toutes les lignes, colonnes et diagonales. Les tests sur les cases individuelles (2 anneaux sur une même case) sont effectués ensuite de manière séparée.

<u>Exemple :</u>

L'IA vérifie s'il y a 2 de ses grands anneaux placés sur la première ligne dans les cases 1 et 2 ou 1 et 3 ou 2 et 3. Puis elle fait de même pour les moyens et petits anneaux. Puis tous les tests dans l'ordre croissant et dans l'ordre décroissant.



Voici donc les différentes étapes réalisées par l'IA dans le programme :

Si 1) n'est pas vérifié, l'IA passe aux vérifications 2) etc...

1) L'IA vérifie si elle peut gagner en analysant toutes les lignes et les colonnes : si elle trouve 2 anneaux qui peuvent former une combinaison gagnante et si la case pour réaliser cette combinaison est vide alors elle place l'anneau dans cette dernière. Ceci est effectué grâce au système plus haut. L'IA remporte ainsi la partie.

2) Cette seconde étape est sensiblement la même que la précédente du point de vue du programme. En effet, il suffit d'établir les mêmes tests que dans 1) mais pour les anneaux de l'adversaire. L'IA analyse donc si son adversaire a une possibilité de gagner. Une fonction ContreStatégique a également été créée pour essayer de contrer tout en jouant un bon coup.

3) Lorsque l'IA ne peut pas gagner au prochain coup et n'a pas à contrer son adversaire, elle analyse les anneaux qu'elle a disposés. Elle analyse de la manière suivante : si elle a posé un anneau et qu'à partir de lui, elle possède 2 cases libres qui pourraient former une combinaison gagnante avec cet anneau;, elle pose un anneau dans l'une de ces 2 cases. On rappelle que grâce aux boucles précédemment évoquées, toutes les manières de gagner sont testées mais l'IA, si elle en a plusieurs, la première qui vérifie les conditions.

4) Si l'IA n'est pas arrivée à remplir l'une des conditions précédentes, elle va effectuer le même type de test encore une fois mais cette fois-ci elle va chercher 3 cases vides qui peuvent former une combinaison gagnante. Et si elle ne peut pas, elle joue un coup au hasard dans l'une des cases libres.

Difficultés rencontrées:

La difficulté principale a été de s'organiser en groupe en prenant en compte les différences de niveau en java pour que tout le monde participe à son niveau et progresse.

Il a été aussi difficile de comprendre toutes les subtilités du jeu afin de les programmer. Nous avons alors joué dans plusieurs configurations pour faciliter le développement des fonctionnalités.

Enfin l'IA à pris pas mal de temps puisqu'il fallait réfléchir à un ordre de priorité dans les coups faits par l'ordinateur. Plein d'améliorations ont été faites au cours du temps pour la rendre pertinente même si elle n'est pas parfaite.

Conclusion:

Ce projet a été bénéfique pour chacun de nous trois car il a éveillé notre curiosité en nous motivant à progresser en programmation pour l'améliorer continuellement. Nous avons de plus appris à nous répartir les tâches et à faire confiance au travail de l'autre tout en s'entraidant car les niveaux étaient différents. De plus, ce projet nous a permis d'acquérir une certaine autonomie nécessaire au métier d'ingénieur. Pour finir nous avons pu voir que la réalisation d'un projet commandé de A à Z passe par de nombreuses étapes différentes de la réflexion à la finalisation.

AUTO ÉVALUATION FINALE

La réalisation de ce projet s'est réalisée en 3 grandes étapes:

- Réflexion commune / Compréhension du jeu
- Développement des algorithmes de test / Premières Versions du Jeu
- Création et amélioration de la base finale du programme Java

Nous avons tous les 3 participé aux 3 étapes de la création mais de manière différente en fonction de notre niveau. De manière générale, chacun à pu faire ce qu'il avait prévu au début du projet.

- Steven a passé beaucoup de temps à développer les algorithmes de test et ensuite le programme final. Une grande partie du programme actuel lui est due.
- Noé a commencé par prévoir les différentes modes de jeu et le mettre en place pour ensuite travailler sur les 2 IA, plus particulièrement la version "HARD"
- Maël à aidé Noé à la mise en place des modes de jeu au début du programme pour ensuite s'occuper de l'affichage final du jeu sur l'interface graphique. Du fait d'un niveau plus faible, les tâches de programmation effectuées sont moins lourde mais il a compensé en mettant à disposition ses compétences en graphisme et également en rédigeant les autoévaluations et le rapport.

Pour conclure sur ce projet, nous avons plutôt bien su tenir compte des écarts de niveau et des compétences de chacun pour produire le meilleur programme possible. Nous avons tous progressé que ce soit dans le travail en groupe ou dans la programmation.