Trinôme : SINKOVICS Adam, TRAN Thibault, SITRUK Timéo STHB7

Compte-rendu projet informatique S1 :

Table des matières

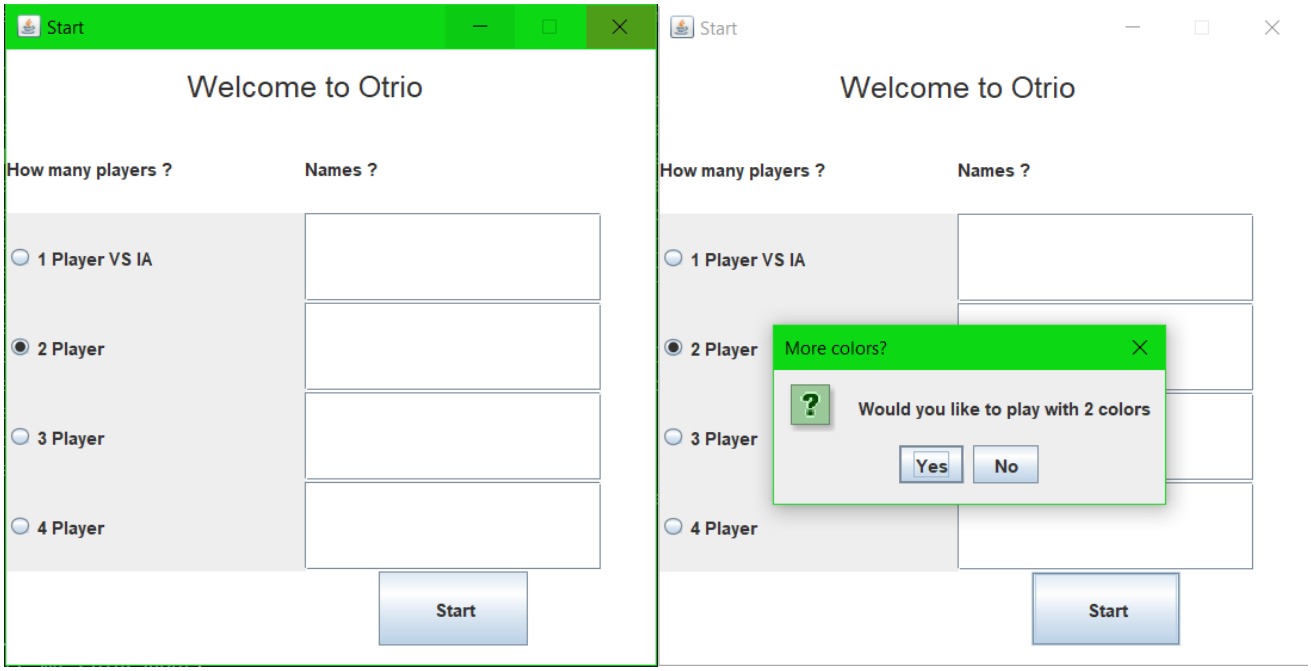
Fonctionnement du Jeu :

A. Coté utilisateur :

A son lancement le jeu nous accueille avec une fenêtre de démarrage. Dans celle-ci nous sommes invités à saisir :

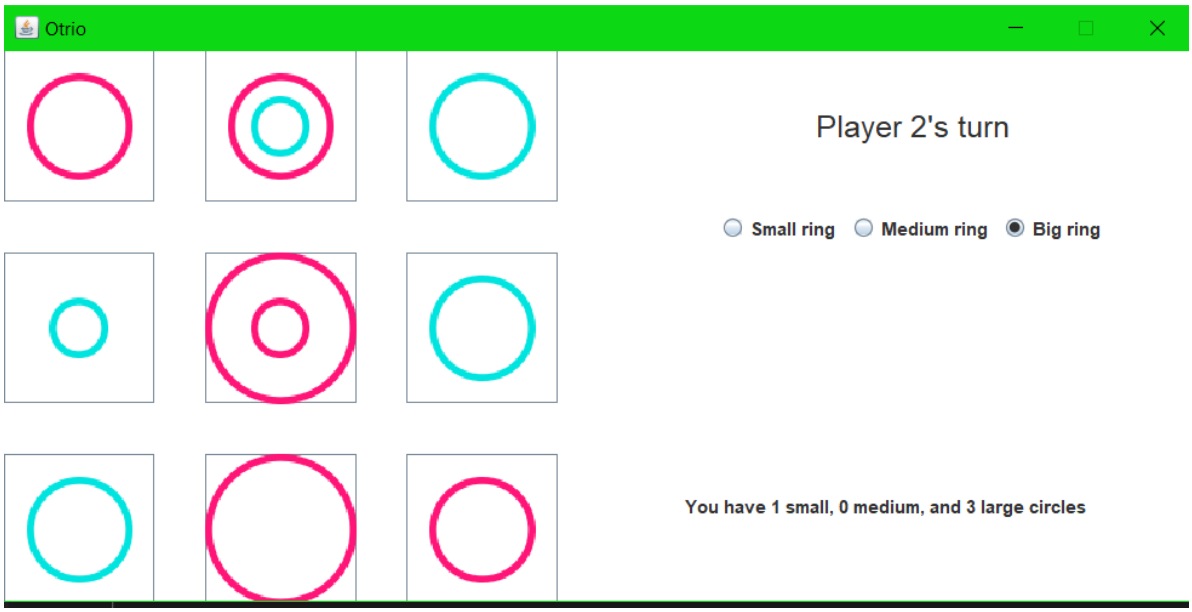
- Le nombre joueur
- Le nom des joueurs.

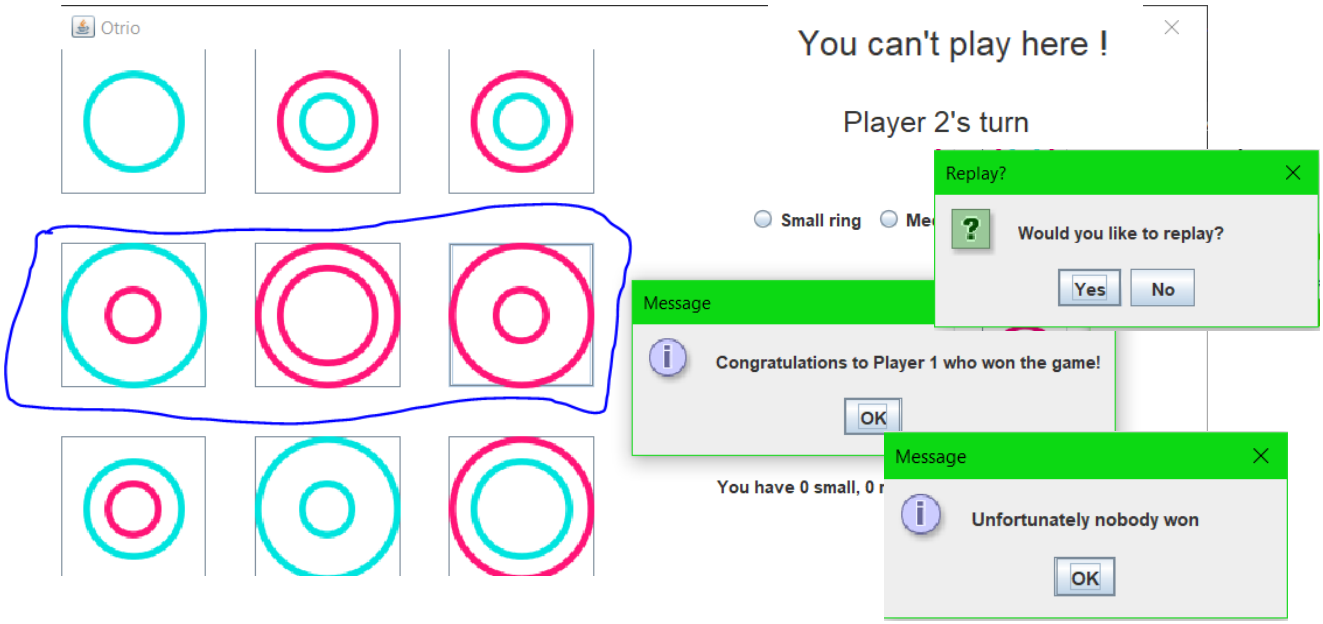
Une fois ces données renseignées, il ne reste plus qu' à cliquer sur le bouton "*Start*" .
Si le joueur sélectionne "*1 Player VS IA"* , il joue automatiquement contre l'ordinateur.
Si les joueurs sélectionnent "*2 Players*", une fenêtre leur demandera s'ils veulent jouer avec une ou 2 couleurs.



Remarque : Si aucun nom n'est saisi, des noms par défaut seront attribués.

Lorsque le bouton *start* est cliqué, la fenêtre de démarrage se ferme.
La fenêtre de jeu s'ouvre alors. Elle est constituée du tableau de jeu interactif à gauche, d'un sélecteur de taille de pion, d'un inventaire, et d'une zone de texte qui affiche des instructions pour le joueur. Le joueur n'a alors plus qu'à se laisser guider et cliquer à l'endroit où il veut placer un pion. Il peut suivre en direct l'évolution de son inventaire, sera informé de si le coup qu'il essaye de jouer est possible ou non. Si un joueur gagne ou que la partie est terminée, les joueurs en seront informés par un pop-up. Il leur sera ensuite demandé s'ils veulent rejouer ou non.





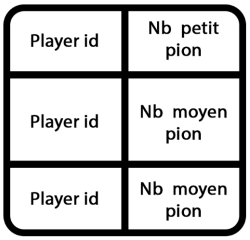
B. Gestion du Jeu :

1. Stockage du tableau de jeu :

Nous avons décidé de stocker toutes les informations du plateau de jeu de la partie en cours (emplacement, taille et couleur des pions placés) dans un tableau tridimensionnelle 3*3*3. Chaque étage du tableau correspond à une taille de pion. Chaque joueur possède un identificateur (un entier entre 1 et 4) qui lui est unique. Lorsque qu'il décide de placer un pion sur le plateau, c'est en fait son identificateur qui est placé dans le tableau 3*3*3 au bon étage, en fonction de la taille de pion choisie..

2. Stockage des informations des joueurs

Pour chaque joueur, une instance de la classe Player.java va être créée. Cette instance va contenir son nom, son identificateur et son inventaire sous forme d'un tableau 3*2 :



C. Gestion et communication avec l'interface graphique :

Nous allons ici détailler l'affichage du plateau de jeu : le plateau de jeu est contenu dans un conteneur (appelé *Panel*). Ce *Panel* possède un layout (méthode de disposition des éléments en son sein) : ici c'est une grille 3 par 3. Chaque case de ce *panel* contient un autre *panel* de type *LayeredPanel* qui permet d'empiler des éléments sur un même emplacement. Chacun des 9 *LayeredPanels* contient 3 *Labels* (objet qui permet de contenir et afficher un élément de type texte ou image) empilés les uns sur les autres ainsi qu'un bouton invisible sur sa couche supérieure. Lorsque l'on veut afficher un pion dans le tableau nous mettons à jour le bon *Label* avec la bonne image.
Nous avons mis en place un certain nombre de fonctions accessibles dans la classe principale pour pouvoir communiquer avec l'interface graphique (envoyer et récupérer des informations). Ces fonctions sont les suivantes :

```
 * name.create() // initialise une fenetre / plateau vierge
 * name.tryPawn(int PlayerId) //attend que le joueur clique sur une case pour pouvoir ensuite utiliser getLastX/Y
 * name.refreshScreen(GameTable, Tableau des joueurs, int PlayerId) : Actualise l'affichage en parcourant tout le tableau de jeu
 * name.clear() // efface graphiquement tous les éléments du plateau
 * name.setDisplayedText(String Text) // Affiche dans la zone de texte un message
 * name.getLastSize() // retourne la taille du dernier pion joué sous forme d'un int
 * name.getLastX() // retourne la coordonnée en X du dernier pion joué sous forme d'un int
 * name.getLastY() // retourne la coordonnée en X du dernier* pion joué sous forme d'un int
 * name.setInventory(String) // actualise le txt de l'inventaire
 * name.getPlayerNumber() // retourne nombre joueurs selectionnés
 * name.getPlayerName(int PlayerId) // retourne le nom du joueur sous forme d'un STRING (id entre 1 et 4 inclus)
 * name.askReplay() //popup qui demande si on veut rejouer et actualise la variable correspondante
 * name.noOneWon(String name) // popup qui dit que personne n'a gagné
 * name.gameWon() //popup qui dit qu'un joueur a gagné
avec name le nom de l'objet de la classe GUI créée
```

Pour une meilleure compréhension de notre interface graphique et de la construction de ses fonctions vous pouvez vous référer aux commentaires de *\src\GUI\GUI.java,* tout y est expliqué.

D. Détection de la victoire :

Pour détecter la victoire d'un joueur nous avons créé la classe "fonction.java" munie d'une fonction *check(int PlayerId)*. A chaque fois qu'un joueur place un nouveau pion, nous exécutons cette fonction. Elle retourne 0 tant que personne n'a gagné et un entier entre 1 et 7 en fonction de la combinaison gagnante utilisée. Cette fonction *check* fonctionne de manière astucieuse et ne parcourt pas bêtement tout le tableau de jeu à 3 dimensions.
Par exemple pour vérifier une diagonale nous vérifions qu'un pion du joueur est présent au centre de la grille, si ce n'est pas le cas, inutile de continuer… Nous appliquons un raisonnement analogue pour les lignes et colonnes. (*cf. commentaires de \src\game\fonction.java, tout y est expliqué en détail*)

E. Déroulement d'une partie :

Dans un premier temps, toutes les variables sont initialisées, le tableau du jeu est rempli de 0, la variable *gagne* est mise sur false : c'est l'initialisation.

Nous rentrons ensuite dans une boucle qui attend que l'utilisateur clique sur le bouton *Start* (après avoir saisi les noms des joueurs et sélectionner le nombre de joueurs).

Une fois ceci fait, le programme sort de cette boucle et récupère les valeurs saisies par l'utilisateur.

Nous rentrons ensuite dans une boucle while dans laquelle nous restons tant que le joueur veut rejouer/jouer. Nous rentrons de nouveau dans boucle imbriquée à la précédente dans laquelle le programme reste jusqu'à ce-que personne n'ai gagné. Dans cette boucle, nous avons une variable *cp* (l'index du joueur actuelle) qui est itérée entre 0 et le nombre de joueur moins un. Cet index *cp* nous permet de récupérer le bon tableau associé au joueur.

Ensuite :

- Si c'est au tour d'un joueur de jouer, on attend qu'il clique sur une des cases sur l'interface graphique (fonction *TryPawn* du GUI). Une fois fait, nous convertissons la position de cette case en coordonnées X, Y et la taille choisie sous la forme d'un entier. (fonctions *getLastX, getLastY, getLastSize* du GUI). Enfin nous actualisons l'affichage (fonction *refreshScreen*)
- Nous connaissons maintenant les coordonnées et la taille du pion que le joueur souhaite placer. Nous vérifions qu'il possède bien le pion correspondant dans son inventaire (fonction *hasElement,* de la classe gest.java), et qu'il est possible de le placer, sinon, on redemande de le placer ailleurs (fonction *setDisplayedText* et *tryPawn*).
- S'il peut placer son pion, le programme place l'identificateur du joueur dans le tableau du jeu (*gameTable*), et enlève un pion de cette taille à son inventaire.
- Le programme vérifie si le pion qui vient d'être placé entraîne la victoire (fonction *check* de la classe fonction.java), si c'est le cas, on sort de la boucle, on ferme la fenêtre puis on demande à l'utilisateur s'il veut refaire une partie, sinon on passe à l'étape suivante.
- S'il est encore possible de placer un pion sur la grille de jeu, nous passons au joueur suivant…
- Si c'est à l'ordinateur de jouer, il choisit aléatoirement des coordonnées et une taille de pion. (et réitère jusqu'à qu'il puisse jouer).