Une image contenant Police, logo, Graphique, texte

Description générée automatiquement

**Rapport du projet d’application**

**Travail réalisé en binôme par :**

* Christ Chadrak MVOUNGOU
* Radia MERABTENE

**Groupe :**

**TD :** A **TP :** 1

**Matières évaluées :**

* Génie Logiciel
* Algorithmique
* Interface Homme-Machine

**Année universitaire :** 2024 -2025

1. **Partie génie Logicielle :**
2. **Réalisation de la délimitation système :**

Nous avons adopté le développement en modules, notre application serait un système à utilisateur unique qui est le joueur adversaire de l’IA lançant la partie du jeu.

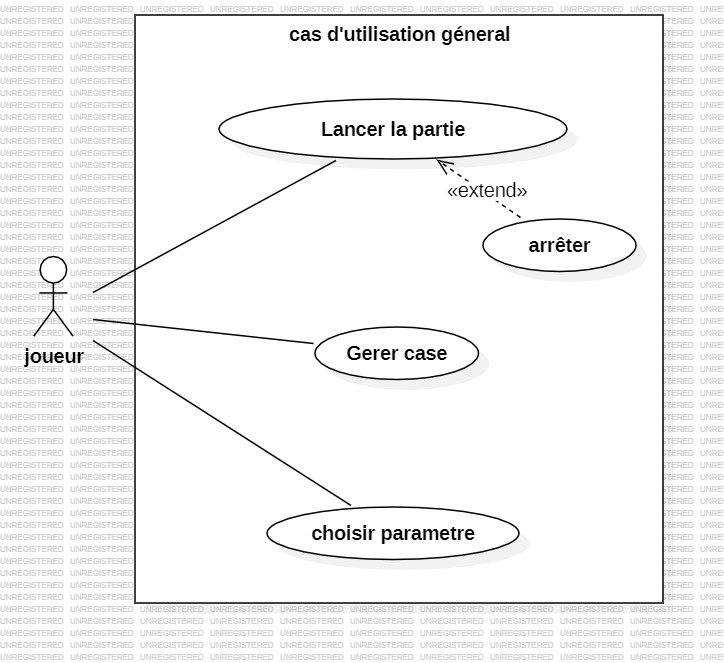
Une image contenant cercle, outil, conception

Description générée automatiquement

1. **Modélisation UML :**

Pour la modélisation nous avons fait le choix de faire deux diagrammes le diagramme des cas d’utilisation et le diagramme de séquences.

1. **Diagramme des cas d’utilisation :**



**Scenario du diagramme des cas d’utilisation :**

Acteurs :

* Principal : Le joueur.

Précondition :

* Une partie n’est pas déjà lancée.

Scénario :

* Le joueur ouvre le jeu, il est dirigé vers la page d’acceuil où il y’aura deux boutons :
  + - Lancer une partie.
    - Paramètre.

1. Premier cas : le joueur clique sur le bouton jouer directement , il aura donc la grille par défaut et commence directement à jouer en considérant les paramètres par défaut( grille 3 x 3, sans cases grises , symbole par défaut x ).
2. Deuxième cas :Le joueur choisit d’abord les paramètres, il valide et la grille se lance selon ses paramètres et il commence à jouer.

Il va alors jusqu’au bout de la partie.

Cas alternatif aux deux cas précédents :

Arrêter la partie : le joueur peut arrêter la partie quand il veut, et le jeu est terminé il est donc redirigé vers l’accueil.

1. **Diagramme de séquence :**

Une image contenant capture d’écran, motif, ligne, art

Description générée automatiquement

1. **Les choix des modules :**

Notre code s’organise en modules en catégories :

Modules abstraits : on trouve dans cette catégorie les modules joueur (Player) qui est un module abstrait qui sera hérité par les deux autres modules plus spécifiques : PlayerHuman qui représente le joueur humain avec comme méthodes des méthodes permettant de gérer les attributs de cette classe.

Et une méthode pour initier le jeu en utilisant le module GameLogic.

PlayerIA : qui représentera l’ordinateur et qui implémentera l’algorithme minmax pour trouver la position la plus avantageuse à jouer pour l’ordinateur.

Modules Actions : ici on trouve principalement le module représentant la logique du jeu qui se nomme GameLogic, ce module contiendra des modules pour vérifier si un des joueurs a remporté la partie (en vérifiant en colonnes en lignes et en diagonales) ou si le match est nul.

Et d’autres fonctions tel la réinitialisation de la partie, jouer une case, création de la grille , redimension de la taille de la grille selon les demandes de l’utilisateur.

Modules Interfaces :

Ici nous mettons le module créant l’interface du jeu.

1. **Convention de codage :**

Pour les conventions de codages nous avons adoptées les mêmes conventions vues en cours à savoir :

**Les variables** : chaque identifiant de variable commencera par la lettre majuscule désignant son type, puis le trigramme correspondant à la classe à qui elle appartient.

**Les types**: chaque type commencera par la lettre T suivi du nom en minuscule.

**Les noms de fonctions**: si la fonction appartient à une classe elle commencera donc par Le trinôme de cette classe sinon elle aura son propre nom commençant par une majuscule.

**Voici le code d’un module modèle suivant cette convention de codage**.

1. **Tests** **unitaires :**

**Les tests unitaires :**

Pour tester notre code nous avons décidé d’analyser la fonction **GLIcheck\_column** : qui va évaluer si une colonne est remplie de façon avantageuse par rapport à un joueur( ce joueur est specifié selon sa couleur passé en parametre) c’est-à-dire si la partie est remportée par un joueur en analysant les colonnes de la grille du jeu.

Fonction choisie : GLIchech\_column.

Signature de la fonction : def GLIcheck\_column(self, iRow: int, iCol: int, oColor: TColor) -> TPlayer | None:

Les variables influentes sur la partie : iRow : l’indice de la colonne à verifier.

iCol : l’indice de la ligne à verifier.

oColor : la couleur du joueur

1. Variable iRow : l’indice de ligne
2. **Valeur :**

**Domaine moyen :** 0 ≤ iRow≤ taille\_max de la grille (3 ou 4 ou 5) -1 **(A)**

Car l’évaluation des indices commencent à partir de zéro.

**Domaine limite :** iRow ≥taille\_max de la grille (3 ou 4 ou 5). **(B)**

1. **Signes :**

iRow est positif **(C)**

iRow est negatif **(D)**

1. Variable iCol : l’indice de la colonne :
2. **Valeur :**

**Domaine moyen :** 0 ≤ iCol≤ taille\_max de la grille (3 ou 4 ou 5) -1 **(E)**

Car l’évaluation des indices commencent à partir de zéro.

**Domaine limite :** iRow ≥taille\_max de la grille (3 ou 4 ou 5). **(F)**

1. **Signes :**

iRow est positif **(G)**

iRow est negatif **(H)**

1. Variable oColor : la couleur du joueur  :
2. **Valeur :**

**Domaine moyen :** Ocolor = { Rouge, Vert, Jaune , Bleu} **(I)**

**Domaine limite :** Ocolor ne prend aucune couleur ou une couleur non sité dans le domaine moyen **(J)**

**L’énumération des jeux de tests :**

**A+C+E+G+I**: exemple de test : dans le cas ou la grille a pour dimension 3 \*3 :

iRow = 2

iCol= 1

Ocolor= Rouge

**B+C+E+G+J :**

iRow= 3

iCol= 2

Ocolor = noir

Erreur car l’utilisateur utilise une couleur en dehors des couleurs permises.

Observation :

Pour plus de sécurité : on doit typer la valeur de icol et irow a un unsigned int au lieu de int pour éviter des valeurs négatives dans les indices.

Ocolor : créer une énumération des couleurs possibles pour faciliter les choix des joueurs.

1. **Partie algorithmique :**
2. **L’Algorithme MinMax :**
3. **Partie IHM :**

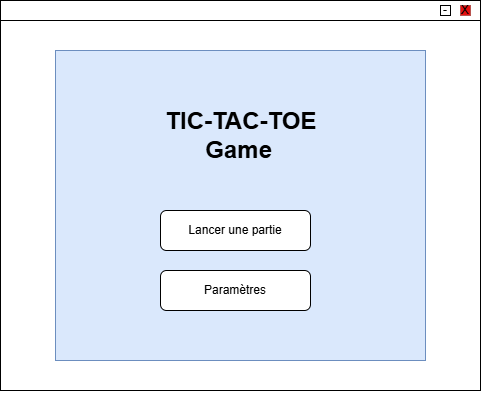
**Interface :**

**La page d’acceuil :**

Quand l’utilisateur/joueur clique sur l’icone du jeu il tombera sur cette fenêtre ou deux boutons essentiels se trouveront :

**Lancer la partie** : si le joueur clique dessus une partie du jeu se lancera avec des paramètres par défaut notamment une grille de jeu 3\*3.

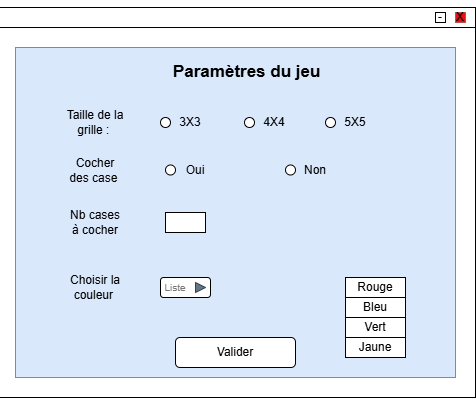
**Paramètre**: si le joueur clique dessus un petit formulaire lui permettant de choisir ses paramètres de jeu s’affichera



**La page des paramètres du jeu :**

Pour le choix des symboles, nous partons sur des jetons pour lesquelles le joueur aura le droit de choisir la couleur. Les couleurs disponibles seront sous forme d’une liste déroulée.

Nous avons également laissé le choix à l’utilisateur de choisir la dimension de sa grille

****

**La grille du jeu :**

D’abord un menu qui va permettre de terminer la partie , un undo( en option ) pour décocher une case (la dernièrement cliquée) et puis un bouton pour réinitialiser une partie et permette de recommencer à nouveau de jouer.

Plus à gauche un frame qui va indiquer à qui parmi les joueurs est le tour de cocher une case, le bouton du joueur concerné change de couleur en vert .

