AVIGNON UNIVERSITÉ

Rapport Projet - Quixo

Étudiant
Michel Marie LAMAH

6 janvier 2023

Master Informatique
Intelligence Artificielle

UE Conception Logicielle
ECUE Application de conception

Responsable
Mikael ROUVIER

UFR
SCIENCES
TECHNOLOGIES
SANTÉ

AVIGNON UNIVERSITÉ
CERI

CENTRE
D'**ENSEIGNEMENT**
ET DE **RECHERCHE**
EN **INFORMATIQUE**
ceri.univ-avignon.fr

Sommaire

1 Choix technologique

1.1 Le language de programmation

Pour le développement de cette application j'ai opté pour l'utilisation du langage de programmation **Python** [1] et celà pour les raisons citées dans le tableau 1 ci-dessous :

Raison	Explication
Facile d'utilisation	En tant que étudiant en IA, le langage Python est celui qu'on utilise le plus, du coup pas de veille technologique à faire dessus
Dispose des structures de données adaptées au projet	Les matrices de la librairie numpy répresente parfaitement la structure du jeu Quixo et Numpy dispose aussi de plusieurs fonctions prédéfinies pour manipuler ces types de données

Table 1. Raisons du choix de Python

A ces raisons j'ajoute aussi mon **envie** de vouloir faire une applications des **design pattern** dans le langage **python**, chose que je n'avais pas fait auparavant.

1.2 Les librairies utilisées

Après avoir choisi le langage de programmation Python, plusieurs librairies s'offraient à moi pour résoudre le problème, le tableau 2 ci-dessous résume ces dernières.

Librairie	Pourquoi
PyGame	Pour développer les interfaces
Numpy	Pour représenter le plateau sous forme de matrice et manipuler ces données
Pydoc	Pour générer les documentation

Table 2. Les librairies utilisées

Pour installer **numpy** [2] il suffit de faire

```
pip install numpy
```

Console 1. Installation Numpy

Pour installer **pygame** [3] il suffit de faire

```
pip install pygame
```

Console 2. Installation PyGame

Quand à **Pydoc**, elle est déjà installée avec python, du coup pas besoin de l'installer

1.3 Éditeur

Pour le développement j'ai utilisé l'éditeur **PyCharm**, un éditeur très facile à prendre en même et fait spécialement pour le langage python et vous permet même d'installer les librairies sans utiliser l'invite de commande grâce à une interface.

1. https://www.python.org/
2. https://numpy.org/install/
3. https://www.pygame.org/wiki/GettingStarted

Elle dispose d'ailleurs une version pour les **éducation**[4] qui contient les mêmes fonctionnalités que la version professionnelle.

2 Fonctionnalités

2.1 Un menu avec plusieurs options

Cette application application dispose d'un menu avec plusieurs options, dont entre autre le **menu principal** :

Figure 1. Main Menu

Ce menu contient **3** sous menu :

1. **Settings**
   Ce sous menu menu contient nous permet de notamment choisir le format du jeu et définir les joueurs

4. https://www.jetbrains.com/fr-fr/pycharm-edu/

Figure 2. Sous menu settings

2. **Play**
   Si tous le format du jeu est choisi et les joueurs défini, ce sous menu nous permet de faire une partie du jeu, comme indiqué à l'image 3 ci-dessous :

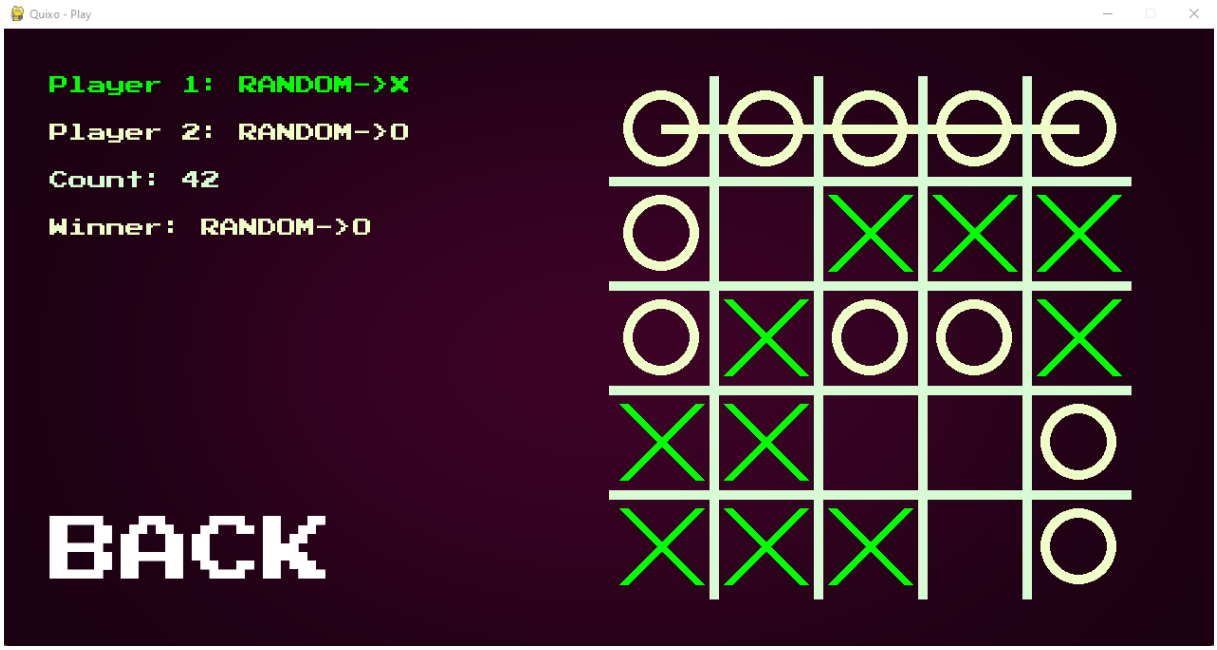

Figure 3. Une partie de jeu

Dans le cas contraire, une fenêtre indiquant ce qu'il faut faire s'affiche, comme indiqué à l'image 4 ci-dessous :

Figure 4. Fenêtre d'erreurs

3. **Quit**
   Pour quitter le jeu

2.2 Plusieurs formats de jeu

En cliquant sur le bouton **Format**, dans le sous menu settings présenté dans l'image 2, une fenêtre s'affiche permettant de choisir le format du jeu, nous disposons de 3 formats : **3x3**, **4x4** et **5x5**

Figure 5. Format de jeu

2.3 Plusieurs modes de jeu

En cliquant sur le bouton **Player 1** ou **Player 2**, dans le sous menu settings présenté dans l'image 2, une fenêtre s'affiche, nous permettant de choisir le type joueurs, nous disposons de 3 type : **AI**, **RANDOM**, **HUMAN**, comme 6indiqué à l'image ci-dessous :

Figure 6. Format de jeu

- **Human** : mode manuel
- **AI** : mode automatique utilisant l'algorithme min-max
- **Random** : mode automatique faisant un choix aléatoire parmi les déplacements possibles

Donc au total, nous disposons de **6** modes possibles :

- **AI** vs **AI**
- **RANDOM** vs **RANDOM**
- **HUMAN** vs **HUMAN**
- **AI** vs **HUMAN**
- **AI** vs **RANDOM**
- **RANDOM** vs **HUMAN**

3 Conception

3.1 Patterns utilisés

Plusieurs patterns on été utilisé pour mieux structurer le projet.

- **Création**
  - **Factory Method** : Pour la création des joueurs
  - **Singleton** : Pour la création des interfaces
  - **Prototype** : Pour cloner les états du jeu, utile pour la création de l'arbre
- **Structure**
  - **MVC** : Architecture du projet
  - **Façade** : Plusieurs interfaces n'étaient accessibles qu'à partir d'une seule interface
- **Comportement**
  - **Observer** : Contrôleur est contenu dans la vue

3.2 Diagramme de classe

Le pattern **Singleton** a été utilisé par toutes les vues, l'image 7 ci-dessous illustre l'un des diagrammes simplifié

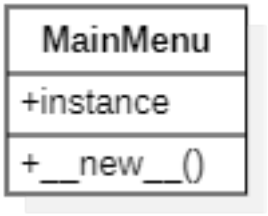

Figure 7. Pattern Singleton

Le pattern **Prototype** n'a été utilisé que la classe State qui nécessitait de faire des copy de l'état du jeu pour chaque noeud de l'arbre, l'image 8 ci-dessous montre une simplification de son diagramme

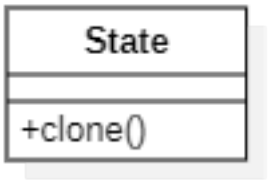

Figure 8. Pattern Prototype

Le pattern **Factory Method** n'était utilisé que pour la création des joueurs, l'image 9 ci-dessous illustre son diagramme simplifié

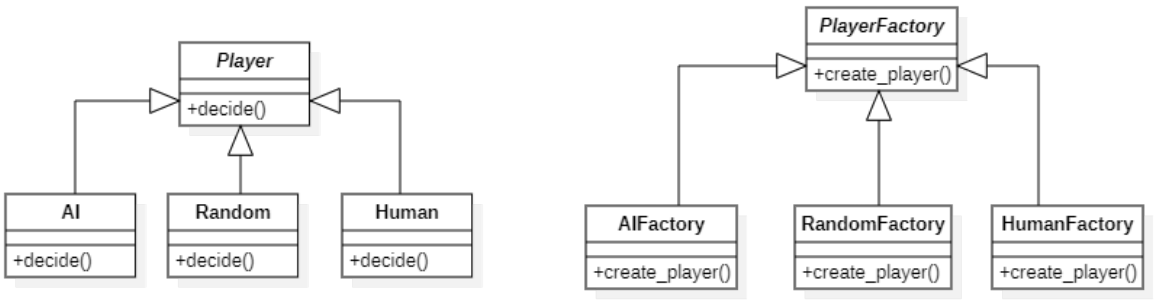

Figure 9. Pattern Factory Method

Le pattern **Façade** n'a été utilisé que par la classe MainMenu qui représentait le sous système formé de toutes les autres interfaces, l'image 10 ci-dessous, nous montre son diagramme

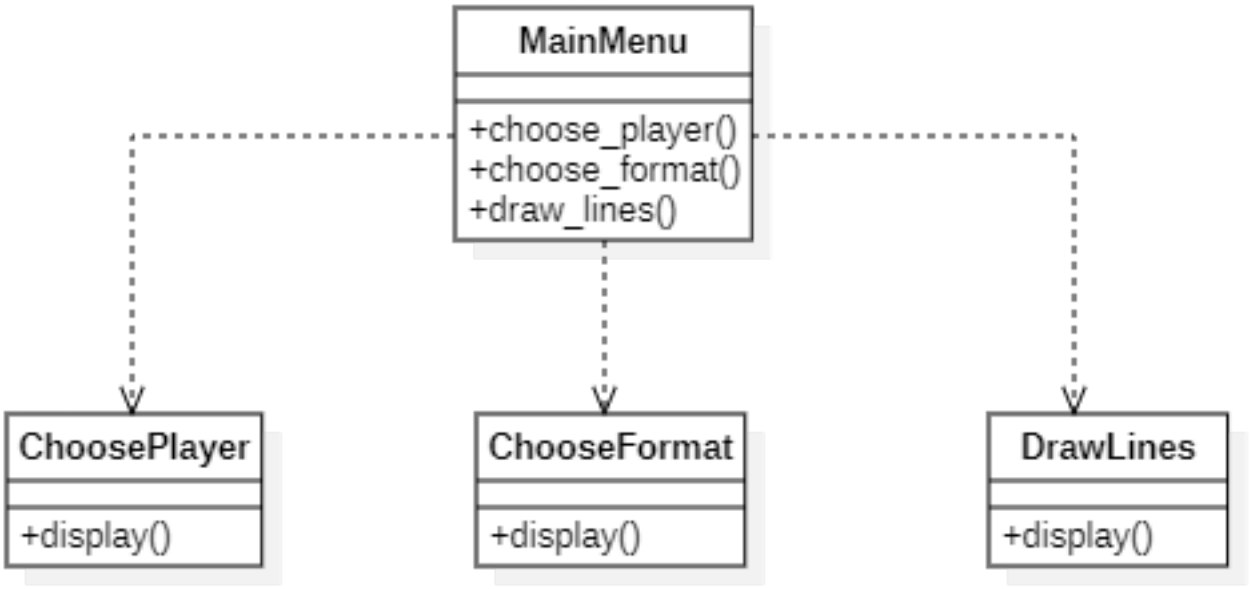

Figure 10. Pattern Façade

Pour le pattern **MVC** a été combiné avec le pattern **Observer**, c'est-à-dire que le contrôleur a été inclus dans la vue et à chaque opération sur la vue, elle appelle directement le contrôleur pour les vérifications, l'image 11 ci-dessous, nous montre une implémentation.

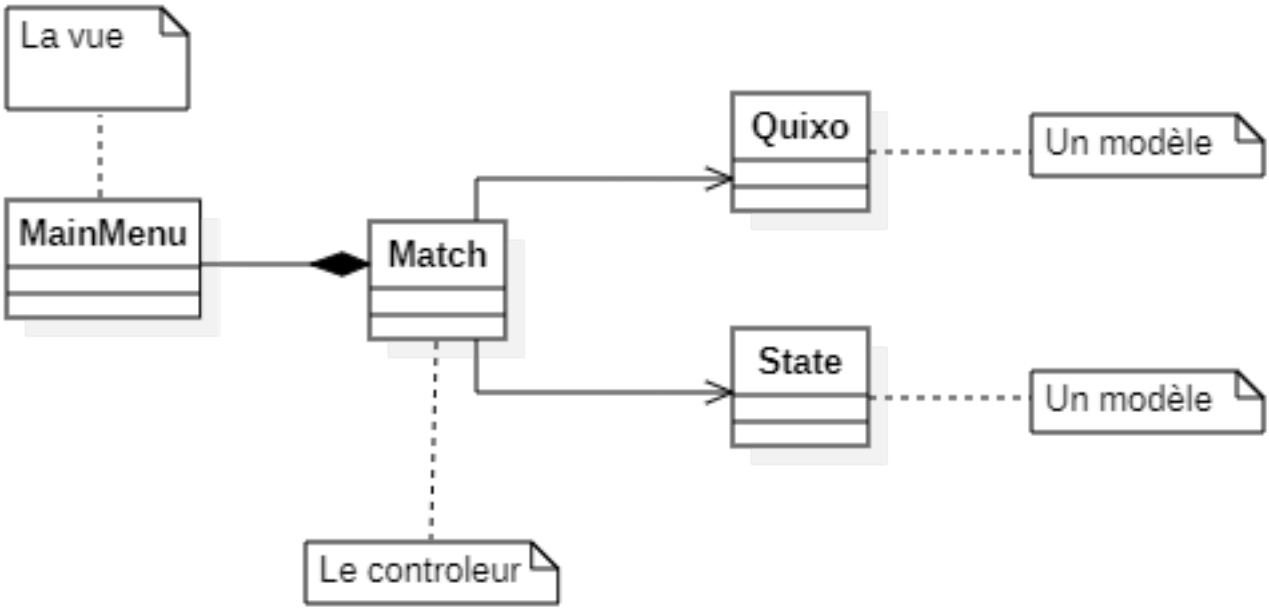

Figure 11. Pattern MVC + Observer

4 Difficultés rencontrées

Durant la réalisation du projet plusieurs difficultés ont été rencontrés dont entre autres :

- **Le choix du langage de programmation**
- **Le choix de la librairie pour les interfaces**
- **La détection des designs patterns**
- **Définir la bonne fonction d'évaluation**
- **Construction de l'arbre**

5 Améliorations possibles

Plusieurs améliorations peuvent être apporté à ce projet dont entre autre :

- Associer une système de stockage au jeu, pour sauvegarder l'historique des parties
- Optimiser un peu le programme car il est un peu lent lorsque l'un des joueurs est l'IA
- Utiliser d'autres librairies pour générer l'arbre du jeu qu'on a créé à la main, par exemple la librairie **Monte Carlo Tree Search**
- Voir s'il y a que code qu'on pourrait refactoriser en pattern, pour encore optimiser
- Offrir la possibilité aux joueurs de choisir sa couleur
- Offrir la possibilité aux joueurs de choisir le symbole à utiliser
- Associer un minuteur au jeu de sorte qu'àprès une certaine période c'est le joueur adverse qui joue
- Permettre de pouvoir spécifier directement dans l'interface si notre algorithme alpha-beta veut maximiser ou minimiser