

Projet jeu à deux joueurs

Sylvain Lamprier

LIP6 – DAPA – MLIA

- Evaluation
- Présentation générale du projet
 - Les jeux Othello et Awélé
 - Etapes du projet
- Plateforme à développer
 - Architecture générale
 - Implémentation en Python
- Site Web Tournoi

Évaluation

- Assiduité / participation
- Examen Partiel sur Machine
- Comptes rendus de projet
 - Rapports intermédiaire et final
- Soutenance
 - Présentation du travail réalisé
 - Epreuve de modification de code

Awélé et Othello

Awélé



- Initialisation

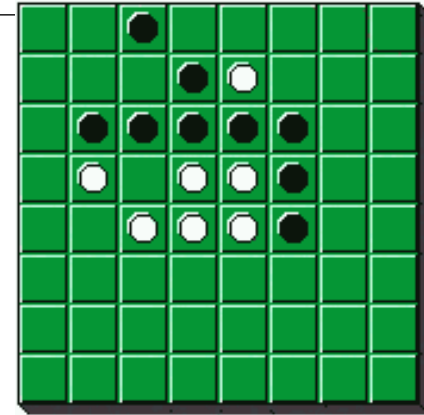
- On répartit 48 graines dans les 12 trous à raison de 4 par trou.
- Les joueurs sont l'un en face de l'autre, avec une rangée devant chaque joueur.
 - Cette rangée sera son camp.

- Les joueurs jouent alternativement.

- Le joueur prend toutes les graines d'un des trous de son camp
- Il les égraine dans toutes les cases qui suivent sur sa rangée puis sur celle de son adversaire suivant le sens anti-horaire
 - une graine dans chaque trou **après** celui où il a récupéré les graines

- Prise :
Si la dernière graine tombe dans un trou du camp adverse et qu'il y a maintenant deux ou trois graines dans ce trou :
 - Le joueur capture ces deux ou trois graines et les met de côté.
 - Puis on itère tant que pas arrêt
 - regarder la case précédente
 - Si elle est dans le camp adverse et contient deux ou trois graines, on récupère ces graines
 - Sinon arrêt
- On ne saute pas de case lorsqu'on égraine
 - sauf lorsqu'on a plus de douze graines, c'est-à-dire qu'on fait un tour complet
 - => on ne remplit pas la case où l'on vient de prendre les graines
- Attention : on ne peut pas manger si le coup qui affame l'adversaire !

Othello



- Initialisation

- Deux pions de chaque couleur sont posés sur les 4 cases centrales.

- Les joueurs jouent à tour de rôle

- En posant un pion
- Jusqu'à ce que les 64 pions soient posés.
- Prise : Si en posant un pion un joueur encadre soit horizontalement, soit verticalement, soit en diagonale, une succession de pions adverses, il les retourne (ils deviennent de sa couleur)

- Le gagnant est alors celui qui possède le plus grand nombre de pions de sa couleur.

Présentation du projet

Etapes du projet

- Semaine 1 à 3:
 - Implémentation Interface de jeu Othello et Awéle
 - Structures de données
 - Actions des joueurs selon règles
 - Implémentation joueurs
 - Joueur humain
 - Joueur aléatoire
 - Joueur premier coup valide
 - Implémentation boucle principale pour une partie
 - Extension pour n parties
 - Statistiques de victoire
 - Initialisation aléatoire
 - Permutation joueurs

Etapes du projet

- Semaine 4:
 - Design d'une fonction d'évaluation des coups
 - Combinaison linéaire de multiples fonctions d'évaluation élémentaires
 - Réglage des paramètres (poids dans la combinaison linéaire)
 - Joueur à horizon 1
 - Joue le coup avec la meilleure évaluation immédiate
 - Evaluation/comparaison du stratège
- Semaine 5:
 - Algorithme de parcours d'arbre
 - Simulation des coups / Modèle adversaire
 - Evaluation des coups à profondeur limitée
 - Algorithme MiniMax

Etapes du projet

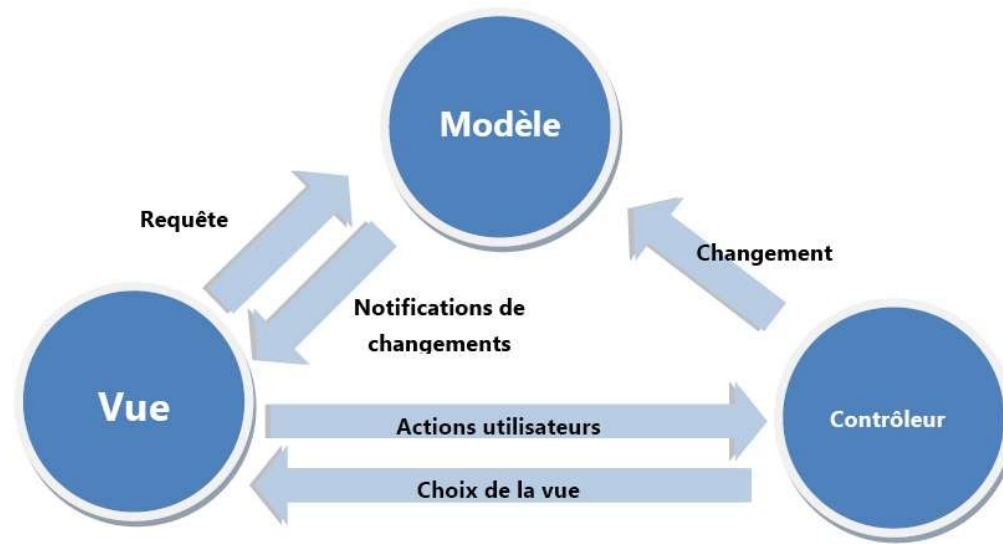
- Semaine 6 et 7:
 - Élagage arbre de recherche
 - Certaines branches de l'arbre peuvent être évitées
 - Accélérer la recherche
 - Augmenter la profondeur
 - Algorithme alpha-beta
 - Optimisation du parcours de l'arbre

Etapes du projet

- Semaines 8 à 10
 - Apprentissage automatique
 - Ce que l'on apprend
 - Fonction d'évaluation
 - Critère d'arrêt d'exploration
 - Différentes techniques d'apprentissage
 - Exploration locale
 - Algorithme génétique
 - Apprentissage supervisé
- Semaine 11
 - Présentation du travail produit
 - Epreuve de modification de code

Architecture générale

- Un exemple de structuration : le modèle MVC

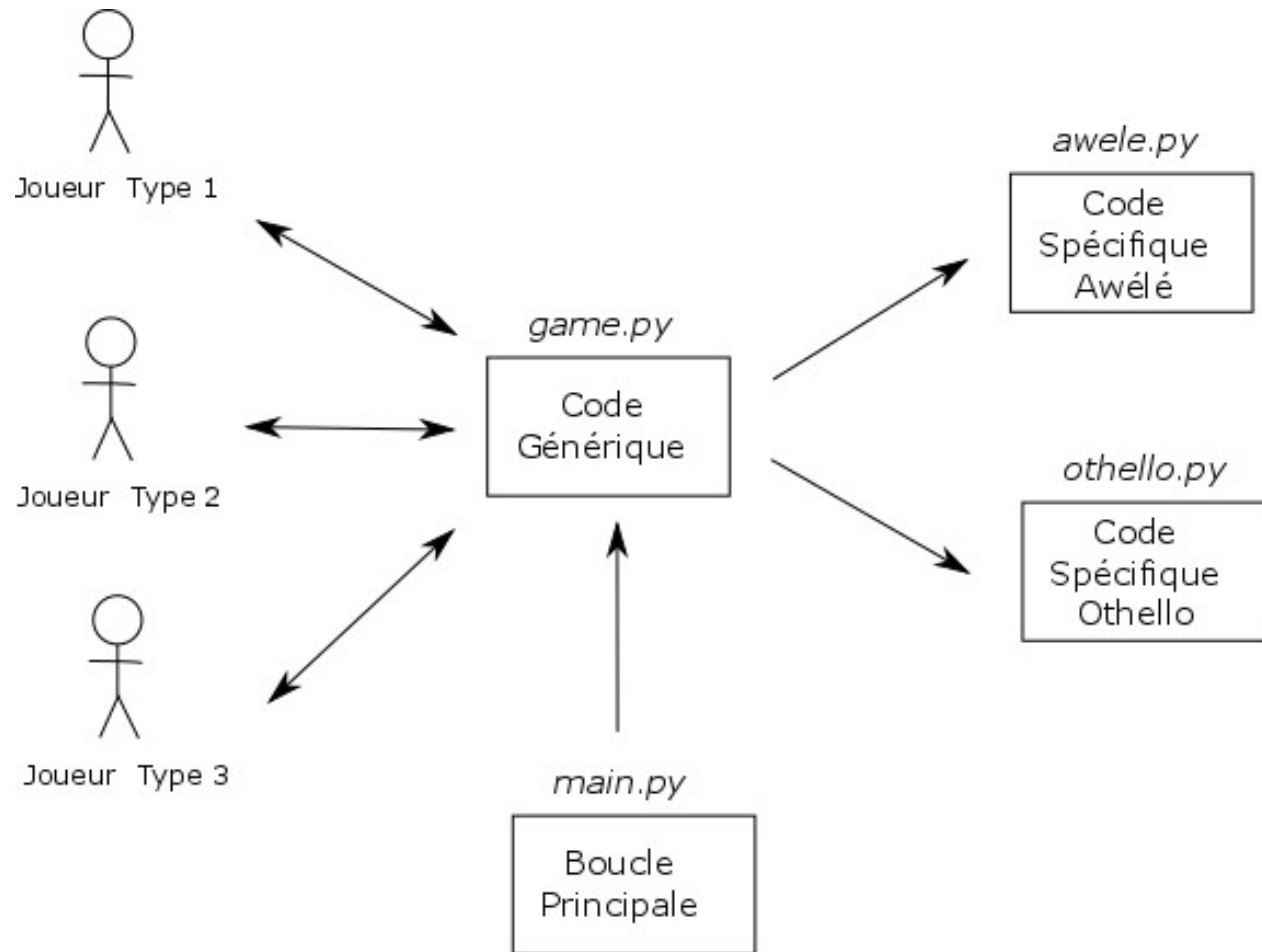


- Permet de bien séparer les différents composants
 - Sécurité
 - Réutilisabilité / Portabilité

Architecture générale

- Programmation par modules Python
 - Pas de modèle MVC mais volonté de bien séparer les choses
 - Cadre général : jeu de plateau
 - On veut développer une implémentation qui puisse rester valide pour tous les jeux de plateau à 2 joueurs possibles (généricité)
 - Décomposition en modules permet d'assurer la généricité d'une partie du code (tout ce qui est commun à l'ensemble des jeux)
 - On souhaite isoler
 - La boucle principale du jeu (déroulement de la partie à 2 joueurs)
 - La structure générique du jeu (tout ce qui peut être commun à l'ensemble des jeux et peut être appelé du module principal)
 - Les spécificités de chaque jeu
 - Le code des joueurs (différentes IA doivent pouvoir s'affronter)

Architecture générale



Code Générique

- Définit les structures de données du jeu

```
# plateau: List[List[nat]]
```

```
# liste de listes (lignes du plateau) d'entiers correspondant aux contenus des cases du plateau de jeu
```

```
# coup: Pair[nat nat]
```

```
# Numero de ligne et numero de colonne de la case correspondante a un coup d'un joueur
```

```
# Jeu
```

```
# jeu:N-UPLET[plateau nat List[coup] List[coup] Pair[nat nat]]
```

```
# Structure de jeu comportant :
```

```
#         - le plateau de jeu
```

```
#         - Le numero du joueur a qui c'est le tour de jouer (1 ou 2)
```

```
#         - La liste des coups possibles pour le joueur a qui c'est le tour de jouer
```

```
#         - La liste des coups joues jusqu'a present dans le jeu
```

```
#         - Une paire de scores correspondant au score du joueur 1 et du score du joueur 2
```


Code Générique

- Interface entre Programme principal et Joueurs
 - Fonction saisieCoup(jeu): jeu => coup
- Interface entre Joueurs et Jeu
 - Fonctions pour décider des coups à Jouer
- Interface entre Programme principal et Jeu
 - Fonctions d'action sur le jeu (réalisation d'un coup)

Contrats

- Lister les fonctions nécessaires pour les différents modules (sans les fonctions privées à un module, uniquement les contrats)
 - Penser au déroulement d'une partie (d'Awélé par exemple)
 - Penser aux besoins de communication entre modules

Programme principal

- Entête du fichier :

```
import sys
sys.path.append("../")
import awele
sys.path.append("../..")
import game
game.game=awele
sys.path.append("../Joueurs")
import joueur_humain
import joueur_random
game.joueur1=joueur_humain
game.joueur2=joueur_random
```

Implémentation

- Donner la boucle principale du jeu
 - Permettant de faire jouer deux joueurs l'un contre l'autre (en supposant implémentées les fonctions énumérées pour chaque module)
 - Quel est le critère d'arrêt
 - Quels affichages ?
 - Quels appels aux fonctions du module game ?
- En TME :
 - Implémenter les fonctions de gestion / affichage du plateau
 - Implémenter le code d'un joueur humain
 - Implémenter les autres fonctions du contrat de game, en supposant données les fonctions du jeu spécifique