Ensta Bretagne
2, rue François Verny
29806 Brest cedex
France

Projet d'informatique

FISE 2024

18 avril 2022

Jeu du tarot Africain

Nicolas Dias
Thomas Prévost

Table des matières

I. Présentation du projet

Nous avons choisi, pour ce projet d'informatique, d'implémenter en Python le jeu du Tarot Africain, qui est un jeu de cartes opposant de deux à quatre joueurs, et demandant stratégie, réflexion mais aussi une part de chance.

1. Règles du jeu et objectifs du projet

Une partie de Tarot Africain se déroule assez simplement. Pour commencer, il faut se munir d'un jeu de tarot, que l'on trie. On place alors d'un côté les atouts (du 1 au 21 ainsi que l'excuse) et d'un autre le reste des cartes séparé et trié par couleur [1].

Début de partie

Donner, à chaque joueur, un paquet trié de cartes de couleur : elles représentent les vies du joueur et sont placées devant lui, visibles de tous les autres joueurs. Les cartes de tarot seront mélangées entre chaque partie.

Déroulement d'une partie

Au début de chaque manche, un joueur désigné distribue 5 cartes d'atout à chaque joueur. Chaque joueur, en commençant par le celui ayant distribué, placera alors un pari sur le nombre de tours qu'il va gagner après avoir consulté ses cartes. Chaque joueur est libre de proposer un nombre de son choix, à l'exception du dernier à parler devant proposer un nombre tel que l'ensemble des paris soit différent du nombre de cartes que les joueurs ont en main (5 au premier tour).

Les joueurs jouent alors à tour de rôle une carte de leur main, en commençant toujours par celui ayant distribué. Gagne le tour le joueur ayant placé la carte la plus forte (à noter que le joueur ayant placé l'excuse choisit sa valeur en fonction de sa stratégie : soit elle est la plus forte, soit elle est la plus faible).

En fin de manche, les joueurs perdent autant de vies qu'il leur manque de tours gagnés pour remplir leur pari (par exemple, un joueur ayant placé un pari de 3 et n'ayant gagné qu'un tour perd $3 - 1 = 2$ vies).

Commence alors la manche suivante, au cours de laquelle seront distribuées 4 cartes (puis 3 à la suivante, puis 2, puis une seule).

Tour à une seule carte

Le tour à une seule carte est particulier : chaque joueur ne regarde pas sa carte, mais celle des autres joueurs. Il dit alors s'il pense avoir une carte plus ou moins forte que celles des autres.

1. pique, carreau...

Fin de partie

Une partie se finit à l'issue de la manche d'une seule carte. À ce moment là, le jeu peut continuer, chaque joueur gardant le nombre de vies qu'il avait à l'issue de la partie précédente. Le joueur ayant distribué transmettant ce rôle à celui à sa gauche.

Le jeu continue alors de partie en partie jusqu'à ce que tous les joueurs sauf un soient éliminés.

2. Pistes envisagées pour l'implémentation

Il a alors été initialement choisi, pour implémenter le jeu de Tarot Africain, le découpage en plusieurs classes : une classe `Joueur`, une classe `Carte`, une classe `Manche` et une classe `Partie`. Structure permettant de manipuler ces différents objets afin de dérouler une partie complète.

Une première structure du programme a alors été schématisée à partir de ces pistes techniques, soulevant alors plusieurs questions au sein du binôme, impliquant finalement de reconsidérer le découpage qui avait été réalisé initialement.

3. Choix techniques

Notre choix, après l'étude préalable ayant été menée, s'est donc porté sur un découpage plus simple que celui initialement prévu : la classe `Carte`, bien trop simple, n'ayant pas de raison d'exister.

Le projet s'articule donc, à ce stade, autour de 3 classes principales :

— La classe `Joueur`, de laquelle sont héritées les classes `JoueurHumain` et `JoueurBot` permettant de faire jouer un humain ou la machine ;
— La classe `Manche` permettant de dérouler une manche de la partie ;
— La classe `Tarot` permettant de dérouler la partie.

II. Le programme sur papier

Le programme construit autour d'un fichier principal contenant le jeu en lui-même, auquel viennent se greffer deux modules, respectivement pour les joueurs et le calcul des coups de la machine.

1. Présentation générale

Le fonctionnement du programme, dans son ensemble, est plutôt simple. On pourra se référer entre autres au diagramme de classes et au diagramme d'actions présentés en annexe.

— L'utilisateur lance une partie par l'exécution de la classe `Tarot`. Il spécifie son nom et le nombre de bots qu'il veut.
— Est alors initialisée la partie, avec le nom des joueurs ; une manche est créée ainsi que les joueurs (humain ou machine) qui sont en lice.
— À chaque tour, un leader est désigné par le programme, et chaque joueur place ses paris et joue. Si le joueur est la machine, son coup est calculé en

fonction des coups des autres joueurs, leurs paris, et adapté en fonction du niveau de difficulté choisi pour la machine ; si le joueur est humain, le programme lui demande son pari et la carte qu'il choisit de poser, connaissant les paris déjà placés et les cartes posées.

— À l'issue de chaque tour est identifié le vainqueur du pli, et à l'issue de chaque manche un vainqueur parmi les joueurs. Les perdants perdent une vie et une nouvelle manche commence.

2. Fonctionnement en détails — jeu contre la machine

III. Le programme en fonctionnement

1. Figures imposées

Les figures imposées suivantes sont vérifiées par le programme :

1. **Factorisation du code** : le programme est composé de trois modules différents responsables de diverses fonctionnalités bien spécifiques, et est donc factorisé ;
2. **Documentation et commentaire du code** : tout naturellement, le programme est intégralement documenté et commenté ;
3. **Tests unitaires** : des tests unitaires ont été écrits pour chaque partie du programme ;
4. **Création d'un type d'objet (classe)** : huit objets ont été créés dans le cadre de ce programme, à savoir `Joueur`, `Manche`, `Tarot`, `Proba`, `Pari1Carte`, `PariMCartes`, `Choix1Carte` et `ChoixMCartes`.
5. **Récursivité** : la récursivité est exploitée dans le jeu de la machine, permettant de pousser plus ou moins la recherche de solution de jeu optimale en fonction de la difficulté choisie par l'utilisateur ;
6. **Héritage entre deux types créés** : les classes `JoueurHumain` et `JoueurBot` héritent toutes deux de la classe `Joueur`.
7. **Héritage depuis un type intégré** : nous avons implémenté une « mémoire », héritée du type `list` de Python, prenant en charge les logs de toutes des différentes manches jouées au cours d'une partie.

2. Tests effectués

3. Limitations observées

Plusieurs limitations ont été identifiées, majoritairment en lien avec le fonctionnement du programme dans la console, sans interface utilisateur. Cette restriction étant à l'origine d'un affichage pas toujours intuitif et clair pour l'utilisateur, qui doit, pour jouer, se contenter d'un affichage textuel.

Malgré tout, en termes d'implémentation uniquement technique, aucune limitation majeure n'a pu être identifiée.

4. Perspectives et améliorations

Ces constatations orientent donc tout naturellement la suite du projet vers l'intégration d'une interface utilisateur permettant d'apporter une solution aux différentes limitations induites par le mode de jeu en textuel, que nous avons pu identifier lors des tests du jeu.

Aucun choix technique n'a encore été fait concernant l'implémentation de l'UI, même si PyQt5 semble être une solution envisageable.

Une autre amélioration envisageable serait également de permettre à l'utilisateur de se créer un profil stocké sur une base de données, lui permettant d'accéder à différentes statistiques concernant ses parties précédentes.

Annexe I. Diagramme de classes

Annexe II. Fonctionnement schématique d'une partie