

UNIVERSITÉ D'AVIGNON
ET DES PAYS DE VAUCLUSE

UNIVERSITÉ D'AVIGNON ET DES PAYS DE VAUCLUSE

RAPPORT - PROJET M1

Jeu de cartes sur Android avec Apprentissage en ligne

Étudiants :

Anthony POUJADE
David LONJON
Cyril CHARBONNEAU
Guillaume BAUDSON

Responsable :

M. Stephane HUET

9 janvier 2016

Table des matières

Introduction	2
1 La Dame de Pique	3
1.1 Règles du jeu	3
1.2 La Cloche de Bois	4
2 L'Apprentissage automatique	5
2.1 Supervisé	5
2.2 Non-supervisé	6
2.3 Online searching	6
2.4 Batch (hors-ligne)	6
2.5 Par renforcement	7
3 Modélisation pour l'apprentissage	8
3.1 TD Gammon	8
3.2 Agent et Environnement	9
3.3 Etats	10
3.4 Système de récompenses et de punition	11
3.5 Librairies utilisées	12
4 Android	13
4.1 Principe de la programmation graphique Android	13
4.2 Pivotement d'écran	13
4.3 Les écrans à prioriser	13
Références	15

Introduction

Le projet "Jeu de cartes pour Android" a été initié en septembre 2015, il se poursuivra jusqu'à juin 2016. Il est dans le cadre de la formation du Master 1 ILSEN (Ingénierie du Logiciel pour la SociétÉ Numérique) au CERI (Centre d'Enseignements et de Recherches en Informatique) université d'Avignon.

L'objectif de ce projet est de réaliser un jeu de cartes (choisi par le groupe d'étudiants), pour la plateforme Android ainsi que l'intelligence artificielle à même de jouer à ce jeu et fonctionnant par apprentissage. Concernant les aspects techniques le logiciel devra pouvoir s'adapter à n'importe quelle taille d'écran. Le moteur du jeu sera réalisé par les étudiants. Le jeu choisi par le groupe d'étudiant est "la dame de pique".

Le logiciel sera codé avec l'IDE* SDK d'android et en java, le langage principal d'android. Le module NDK* permettra de coder certaines parties du logiciel en c++ afin d'obtenir de meilleures performances pour les parties qui en auront besoin. La partie graphique se fera à l'aide de fichier xml et d'images matricielle(jpeg, png), le format d'image se choisira en fonction du besoin de transparence. Les fichiers xml seront compilés dans le code donc il ne sera pas possible de les générer dynamiquement en fonction de la taille décran de l'appareil hôte. La solution consistera à prédéfinir les tailles d'écran les plus utilisés via les fichiers xml et éventuellement par des mêmes images de tailles différentes.

Pour la partie algorithmique, des bibliothèques java seront utilisées pour aider à appliquer l'IA par apprentissage. Pour parvenir à réaliser ce type d'apprentissage il est nécessaire de définir des états, afin d'aider l'IA à identifier des situations similaires passées, et ainsi jouer de la meilleure façon possible. L'IA commencera par jouer mal et progressera au fil du temps. Ces expériences seront stockées via une structure de données, l'IA viendra identifier les similitudes d'un état existant, si ce dernier existe. Dans le cas contraire il ajoutera la situation dans la structure de données et y insérera toutes les connaissances qu'elle possèdera à ce moment là.

1 La Dame de Pique

1.1 Règles du jeu

La Dame de Pique est un jeu de cartes (52) avec quatre joueurs. On répartit les cartes entre chaque joueur, soit 13 cartes.

Avant chaque tour, chacun des joueurs choisit trois cartes de son jeu qu'il donne à son voisin de gauche, puis au tour suivant à son voisin de droite, puis au tour suivant au joueur en face de lui, et enfin au quatrième tour à lui-même (alors, aucun échange de carte n'a lieu). Ces échanges se reproduisent sur le même modèle à chaque série de quatre tours jusqu'à la fin de la partie.

Les échanges terminés, le joueur qui possède le 2 de Trèfle commence la partie.



FIGURE 1 – Exemple du jeu de cartes

Chacun joue ensuite dans le sens des aiguilles d'une montre. Chaque joueur est obligé de jouer la couleur demandée. S'il n'en a pas, il se défausse d'une carte dans une autre couleur. Celui qui a joué la carte la plus forte (du 2 à l'as) dans la couleur demandée remporte le pli et entame le suivant.

Au premier tour, il est interdit de se défausser d'une carte qui vaut des points : cœur ou dame de pique (sauf si le joueur n'a que ces cartes en main). Par ailleurs, il est également interdit d'entamer un tour avec un cœur si personne n'a encore défaussé de cœur lors d'un tour précédent (sauf s'il ne reste que des coeurs en main). À la fin des treize plis, le nombre de points de chaque joueur est décompté. Chaque cœur vaut un point, et la dame de pique en vaut 13. Ensuite, chaque joueur reporte son nombre de points sur le tableau des scores et une nouvelle manche commence.

La partie s'achève lorsqu'un joueur obtient un total de 100 points ou plus, le vainqueur étant celui qui a le moins de points. Il peut arriver que deux joueurs aient le même nombre de points, dans ce cas-là, c'est aux joueurs de se départager ou de laisser le résultat tel quel. La fosse n'est pas représentée dans l'interface de l'application mais est tout de même présente dans nos états.

Pour résumer :

- Chaque joueur possède 13 cartes au début d'une manche ;
- Le but du jeu est ici de totaliser le moins de points possible ;
- Le calcul des points s'effectue ainsi ;
 - Dame de pique : 13 points ;
 - Cartes de la famille des piques : 0 points (sauf la dame, donc) ;
 - Cartes de la famille des coeurs : chaque carte vaut 1 point ;
 - Cartes de la famille des carreaux : 0 points ;
 - trèfles : 0 points ;
- Les points sont enregistrés dans un tableau des scores avant le début de la prochaine manche ;
- Le jeu se déroule en plusieurs manches, jusqu'à ce qu'un joueur atteigne 100 points, le gagnant étant à cet instant le joueur ayant le moins de points ;
- La valeur des cartes est croissante et va du 2 (carte la plus faible) à l'As (carte la plus forte) ;

1.2 La Cloche de Bois

Le but du jeu est donc normalement de se défausser de ses cartes valant des points, et de se défausser des cartes les plus fortes (figures). Néanmoins, si un joueur ramasse tous les coeurs et la Dame de Pique, il inflige 26 points à ses adversaires et ne reçoit aucun point (d'où le terme de "volte", retournement).

On dit parfois d'un joueur qui a réussi une volte ou Grand Chelem qu'il a « déménagé à la cloche de bois ».

Cette tactique est envisageable si le joueur détient de très nombreuses cartes dans une couleur, ou s'il possède de nombreuses cartes fortes y compris à cœur et à pique. Dans ce type de jeu, le joueur se défausse de ses cartes les plus faibles avant que le tour ne commence et dans les premiers tours. Il ne peut néanmoins jouer de coeurs tant qu'un joueur ne s'en est pas défaussé ou joue cette couleur parce qu'il ne lui en reste aucune autre. Si une seule carte valant des points lui échappe, le chelem échoue. Dans ce but, si un joueur est sur le point de réaliser la volte ou grand chelem, certains joueurs essaieront de se sacrifier en prenant volontairement un ou plusieurs coeurs.

2 L'Apprentissage automatique

Pour fournir à un système/ordinateur une manière de percevoir leur environnement et interagir avec lui, un sous-domaine de l'informatique et plus particulièrement de l'intelligence artificielle est consacré pour référencer et indiquer les différents moyens permettant à une machine (au sens large) d'évoluer.

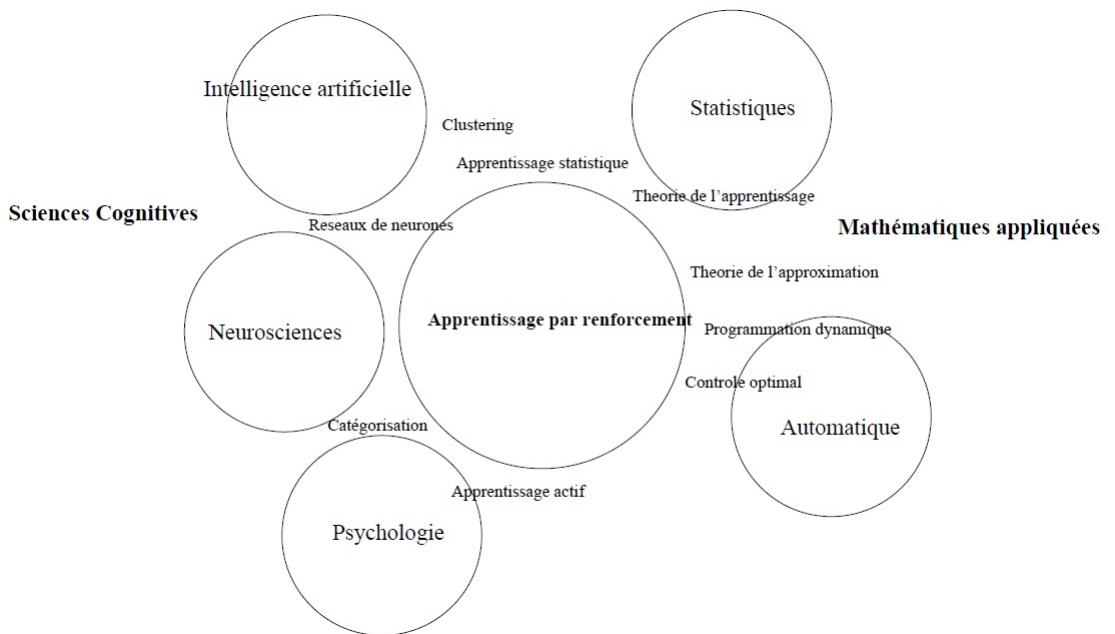


FIGURE 2 – Positionnement de l'apprentissage vis-à-vis des autres sciences

Plusieurs "familles" existent, visant à spécifier la manière de procéder dans le cadre de cet apprentissage.

2.1 Supervisé

On évoque l'apprentissage supervisé si l'on peut classer les informations, les étiqueter. On fournit ainsi au système un certain nombre d'exemples déjà placés dans des catégories et l'on attend en retour la production automatique de règles de la part de la machine à partir de cette base de données.

Pour placer cela dans un cadre concret, cela peut permettre de catégoriser de nouveaux patients au vu de leurs analyses médicales par rapport à des points communs avec les symptômes d'autres patients connus (fournis en exemples). L'apprentissage peut également être semi-supervisé ou partiellement supervisé : soit dans le cas où certaines données sont manquantes,

ou alors que l'étiquetage de données est partiel (qu'une donnée par exemple n'appartient pas à une classe A, mais peut appartenir à une classe B ou C).

2.2 Non-supervisé

L'autre méthode, à contrario de la première énoncée, est l'apprentissage dit non supervisé. Dans ce cas-là, le programme n'a pas de « professeur ». L'algorithme doit découvrir par lui-même la structure des données qu'on lui communique ; il doit cibler les informations et les classer en groupe.

Comme analogie, on peut prendre le cas d'un enfant et lui présenter des mammifères ; c'est à l'enfant de parvenir à faire la distinction entre un reptile, un oiseau et un poisson. Il y a un processus d'apprentissage. L'apprentissage non-supervisé vise donc à caractériser la distribution des données et les relations entre les variables.

2.3 Online searching

L'apprentissage en ligne (online searching) correspond au cas où les données peuvent arriver au cours du temps, par exemple l'état d'un stock où il y a des mises à jour régulières. On dresse ainsi une cartographie des informations qui évolue à chaque nouvelle arrivée.

L'avantage est à double, à la fois cela permet donc de traiter les problèmes qui sont liés au temps, mais également de gérer des applications ayant une masse de données très conséquente ; elle est alors traitée progressivement. Cette méthode permet un apprentissage rapide mais peut entraîner de grandes instabilités.

2.4 Batch (hors-ligne)

La différence fondamentale entre le online searching et le batch (l'apprentissage hors-ligne) tient au fait que dans le premier cas les données arrivent au fur et à mesure alors que dans le second il n'y a qu'une seule entrée de données, au lancement du programme.

Le batch utilise donc une base de données statique, à l'inverse du online qui s'appuie sur quelque chose de dynamique. Pour prendre un exemple et ainsi mieux délimiter les deux types, il y a l'idée des mails. On peut recevoir des mails au fil du temps et les envoyer à chaque fois au système qui s'occupera de les classifier : c'est de l'apprentissage en ligne.

On peut prendre un grand nombre de ceux-ci, les étiqueter et les passer à un système en charge de les classifier : c'est de l'apprentissage hors-ligne. Le batch permet une relative stabilité.

2.5 Par renforcement

L'apprentissage par renforcement consiste pour un système à apprendre, à partir d'expériences, quelle action il doit faire suivant une situation donnée de manière à maximiser la somme des récompenses perçues.

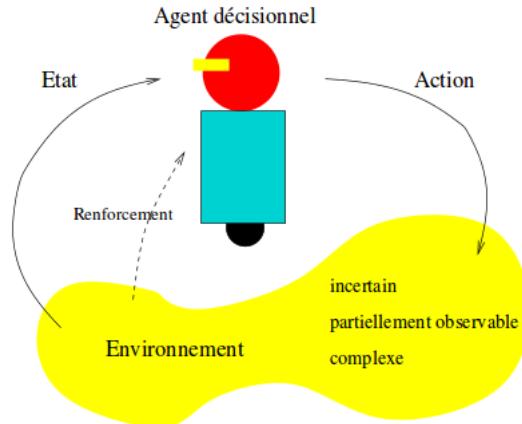


FIGURE 3 – Descriptif de l'apprentissage par renforcement

L'agent doit prendre une décision en fonction de l'état courant ; en retour de son action, l'environnement donne une récompense (positive ou négative). A partir de cela, l'agent doit comprendre qu'est-ce qu'il a fait pour obtenir cette récompense. Il apprend donc au final par ses interactions avec l'environnement et en observant les résultats sur les états de l'environnement. Dans le cadre de ce projet, il s'agit de l'apprentissage le plus proche de notre objectif.

3 Modélisation pour l'apprentissage

3.1 TD Gammon

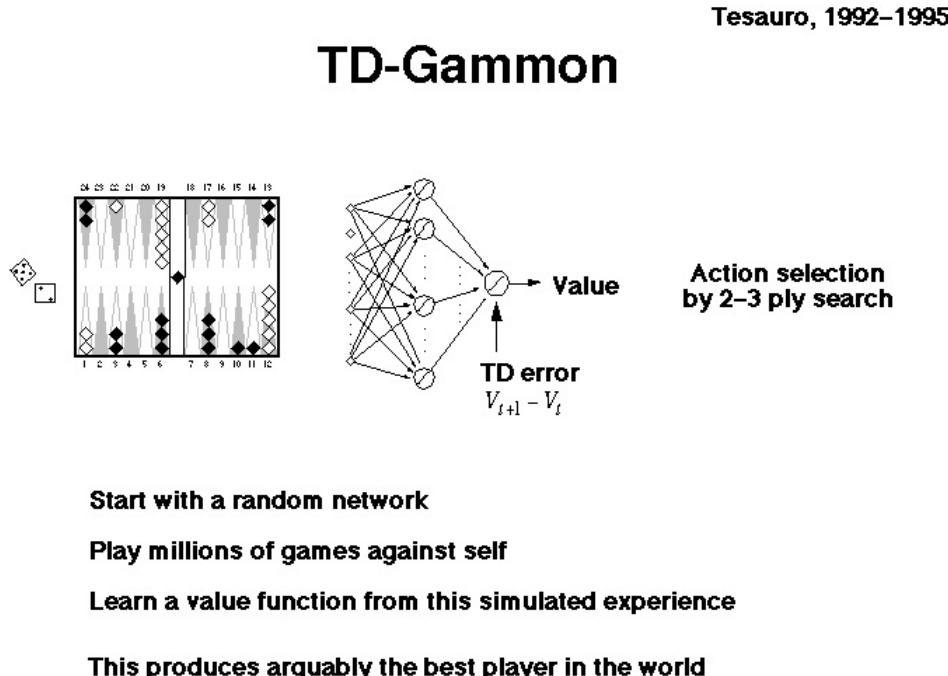


FIGURE 4 – Résumé du TD Gammon

Pour relier la théorie à la pratique, dans les années 1990, de nombreux projets/programmes ont été élaborés pour essayer de simuler une intelligence artificielle et lui fournir les outils pour apprendre, jouer et gagner. L'exemple le plus célèbre est Deep Blue, l'ordinateur qui a affronté le champion d'échecs Kasparov, perdant face à lui en 1996 avant de gagner l'année suivante. Un autre projet, moins connu mais tout aussi prestigieux, intervient en 1992 avec le TD Backgammon, une intelligence artificielle à même de jouer au backgammon. Sa particularité, inscrit dans son nom, est d'utiliser un réseau neuronal artificiel en lien avec l'apprentissage par différence temporelle (en anglais :temporal-difference learning). Grâce à cette méthode, le TD-Gammon a atteint un niveau de jeu légèrement inférieur à celui des meilleurs joueurs de backgammon de l'époque. Il a exploré des stratégies que ces derniers n'avaient pas poursuivis et ont aboutis à des progrès dans la théorie du jeu de backgammon.

Son mode de fonctionnement n'est pourtant pas très différent des autres programmes carte-jeu d'ordinateurs. Chaque tour tout en jouant un jeu, TD-Gammon examine tous les coups légaux possibles et toutes leurs réponses

possibles et choisit le passage qui mène à la position qui a obtenu le score le plus élevé. L'innovation de TD-Gammon est dans la façon dont il a appris sa fonction d'évaluation.

L'algorithme d'apprentissage de la TD-Gammon consiste à mettre à jour les poids dans son réseau neuronal après chaque tour pour réduire la différence entre l'évaluation de l'état du jeu des tours précédents et de son évaluation du jeu actuel ; donc ce qu'on appelle l'apprentissage par différence temporelle. Le score de toutes les positions du conseil du plateau de jeu est un ensemble de quatre nombres reflétant la probabilité de chaque résultat de jeu possible du programme : Blanc gagne normalement, Noir gagne normalement, Blanc gagne un Gammon, Noir gagne un Gammon. Pour la position finale du jeu, l'algorithme compare avec le résultat réel du jeu plutôt que sa propre évaluation de la position du plateau.

Au final, grâce à ses recherches, le TD-Gammon a découvert un coup plus optimal pour débuter une partie et très vite la communauté des joueurs a au final adopté ce premier mouvement.

3.2 Agent et Environnement

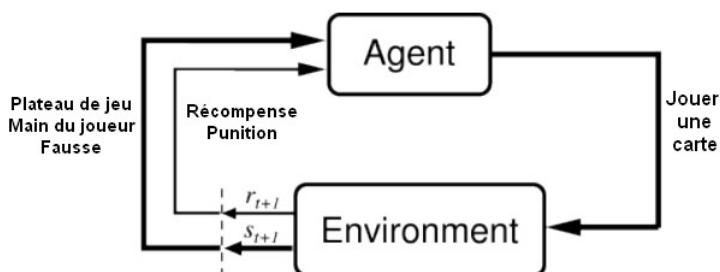


FIGURE 5 – Interaction Agent Environnement

Le fonctionnement donc du TD-Gammon est basé sur l'apprentissage par renforcement qui consiste à considérer un agent autonome plongé au sein d'un environnement. L'agent sera à la fois décideur et apprenant, en effet il interagira constamment avec l'environnement et devra prendre des décisions en fonction de son état courant le tout sous forme d'actions, une action dans le cadre de notre projet étant de jouer une carte de notre main sur le plateau.

L'environnement devra alors répondre à ces actions en présentant de nouvelles situations à l'agent, c'est à dire un nouvel état qui a été modifié par l'action précédemment effectuée. L'agent devra aussi apprendre de ses actions, l'environnement se chargera alors de retourner une récompense sous forme numérique, que l'agent cherchera au fil du temps et à travers ses ex-

périence à maximiser. Entre autre il cherchera un comportement décisionnel optimal, appelé politique ou stratégie, qui est une fonction qui associe l'état courant à une action à exécuter.

3.3 Etats

Dans le cas de notre application les états seront représentés par trois éléments.

- La main du joueur (les cartes qu'il a en main), composé d'un nombre de cartes, les cartes ayant des couleurs et des nombres ;
- L'état du plateau de jeu, c'est à dire les cartes qui ont été jouées pendant le tour et le nombre d'emplacements (4 joueurs 4 cartes potentiellement jouable) ;
- L'état de la fosse , c'est à dire toutes les cartes qui ont été jouées depuis le début de la partie. La fosse n'est pas représentée dans l'interface de l'application mais est tout de même présente dans nos états.

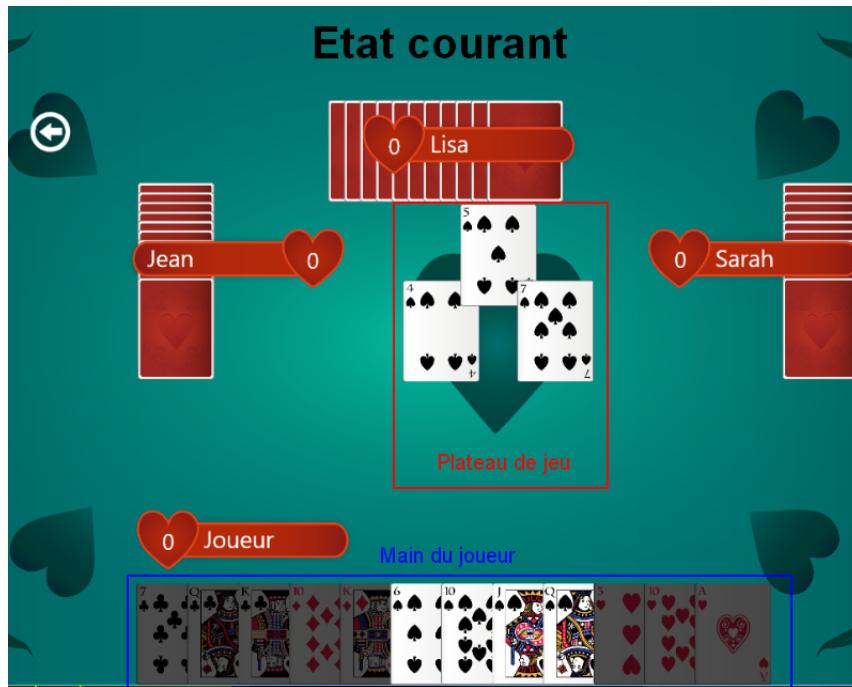


FIGURE 6 – Etat 1

Voici un exemple concret de nos états, on peut voir la main du joueur avec les cartes qu'elle contient, l'état du plateau de jeu ou l'on peut voir 3 cartes posées car nous sommes le 4ème joueur à jouer ce tour-ci, ainsi qu'une fosse qui n'est pas représentée sur l'interface de jeu mais qui représente toutes les cartes qui ont été posées et qui donc ne peuvent ni être présent dans les

mains des joueurs ni sur le plateau.

L'action sera alors faite en fonction de l'état courant, les cartes jouées sur le plateau sont des piques donc nous ne pouvons poser potentiellement que du pique si nous en avons en main ce qui est le cas. Une fois l'action exécutée (jouer du pique) l'environnement retournera une récompense ou une punition, ainsi qu'un nouvel état.

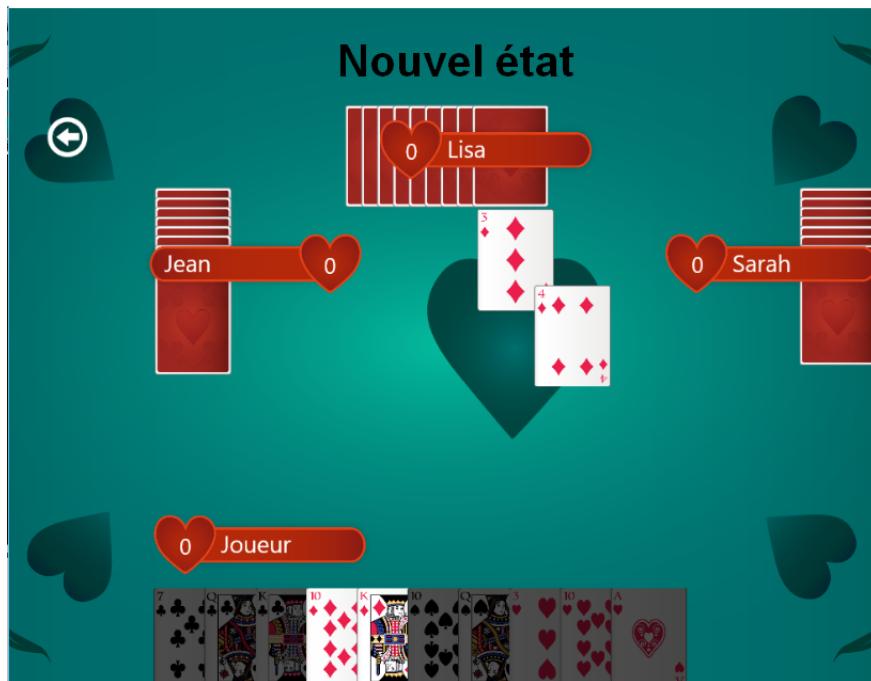


FIGURE 7 – Etat 2

Voici l'exemple de notre nouvel état que nous a renvoyé l'environnement, le nombre de cartes dans nos mains à changer en fonction de l'action effectuée, le plateau de jeu a aussi changé, et le nombre de cartes dans la fosse a augmenté. Ce nouvel état deviendra donc l'état courant de la prochaine action.

3.4 Système de récompenses et de punition

Dans le cadre de notre projet l'environnement récompensera l'agent à la fin de chaque manche lorsque les 4 joueurs auront posés leur carte. La récompense sera choisie en fonction du résultat de cette manche. Une récompense est une valeur numérique négative, positive ou nulle suivant si on veut récompenser ou punir le joueur. Si le joueur au cours de cette manche n'a pas récupéré de carte il sera récompensé au moyen d'un nombre de point important. Si celui ci récupère des cartes, mais que les cartes n'ont pas d'influence

sur le nombre de points récupérés (c'est à dire carreau, trèfle , ou pique en dehors de la dame) alors il sera moyennement récompensé.

Pour ce qui est des punitions lorsque le joueur récupère une carte cœur, l'événement enverra une récompense numérique négative assez élevée. Dans le cas où le joueur récupérera une dame de pique le joueur recevra une punition équivalente à 13 fois la valeur numérique d'une carte cœur.

Concernant la règle de la cloche de bois énoncée plus haut (à savoir qu'obtenir toutes les "mauvaises cartes" du jeu entraînent un fort bonus et un malus aux autres joueurs), elle ne sera pas implémentée dans la première version et sera davantage un objectif bonus ; en effet, on ne peut pas avoir à la fois une vérification à chaque tour puis une à la toute fin de la partie qui en viendrait à inverser toute la tendance, une suite de pénalités régulières qui se conclue par un immense bonus.

3.5 Librairies utilisées

Pour permettre au projet de fonctionner sans devoir partir de zéro, des librairies dans le langage de programmation java seront utilisées pour permettre d'implémenter l'apprentissage par renforcement. La plupart des éléments retrouvés sont issues de projets menés par des universités et qui les mettent ensuite à la disposition d'autres chercheurs/étudiants. De nombreux librairies correspondent à nos attentes mais particulièrement deux d'entre elles :

- The Brown-UMBC Reinforcement Learning and Planning (BURLAP)
- The YOrk Reinforcement Learning Library (YORLL)

Toutes les deux sont disponibles gratuitement, ont été mises à jour récemment et possèdent les composants pour mettre en place l'apprentissage par renforcement. La première, BURLAP, semble un peu plus compliqué à faire fonctionner mais dispose d'une fonction (LearningRate) permettant de modéliser plus simplement la courbe de progression de l'apprentissage de l'IA. Quant à YORLL, le code est bien plus évident avec notamment la présence de classes en java portant les noms de ce qui a été évoqué plus haut, à savoir l'agent, l'environnement ou encore le state. Par ailleurs, il y a également la présence de SARSA, désignant le principe de State-Action-Reward-Action. Nos efforts se tourneront d'abord sur la librairie YORLL et en dernier recours sur celle de BURLAP.

4 Android

Afin de réaliser la partie graphique nous allons utiliser androïd studio, outil développé par google après le rachat d'Android et embarquant l'ensemble des outils de développement qui nous seront utiles.

Android studio possède des outils de génération graphique, un éditeur de texte (code et xml) ainsi que tous les outils de navigation dans un projet. Pour finir android studio possède un émulateur de terminal qui nous permettra de tester les différentes interfaces graphiques en fonction des écrans.

4.1 Principe de la programmation graphique Android

Le principal problème des interfaces graphiques sous Android est qu'il n'existe pas de norme pour les écrans, ainsi une multitude de résolutions est apparue, on en compte plus d'une vingtaine de nos jours. Pour pallier à ce problème Android a mis en place un principe de programmation d'interface graphique à partir de fichiers xml, un fichier xml correspondant à une résolution d'écran. Ainsi pour correspondre à un maximum d'utilisateur il faut créer un grand nombre de fichier xml.

4.2 Pivotement d'écran

Implémenté dans la plupart des smartphone moderne la fonction de pivotement permet de passer d'une vue en portrait à une vue paysage. L'implémentation de chaque vue se fait via un layout¹ spécifique et l'intégralité de l'interface graphique doit y être adapté, par la suite le système s'adaptera automatiquement en fonction de l'inclinaison du téléphone.

Cependant, il est possible de forcer un mode (portrait ou paysage) afin d'assurer une expérience d'utilisation plus simple en fonction de l'application. Dans notre cas, le blocage en mode paysage assurera une parfaite lisibilité de la main du joueur tandis qu'un mode portrait rendrait cette lecture difficile.

4.3 Les écrans à prioriser

Comme vu précédemment la gamme d'écran des smartphones utilisant Android est relativement élevé, il ne sera donc probablement pas possible de s'adapter à absolument tous les systèmes. Cependant, en priorisent les interfaces en fonction du nombre de smartphone en circulation il est possible de toucher un maximum d'utilisateur. Par exemple, et comme le montre le

1. balise de positionnement

graphique suivant, il est possible de toucher plus de 50% de la population avec seulement trois interfaces/fichiers xml.

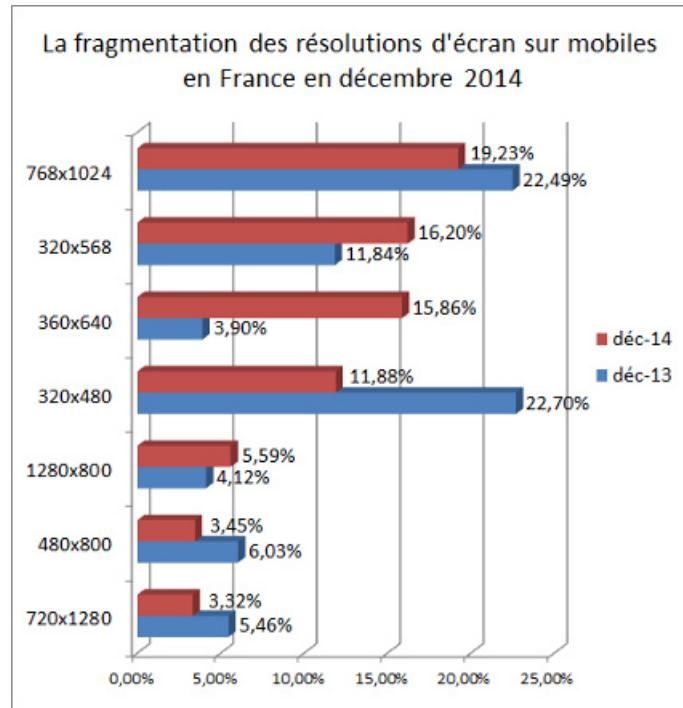


FIGURE 8 – Statistiques

Références

- [1] Andrew G. Barto Richard S. Sutton. *Reinforcement Learning : An Introduction (Adaptive Computation and Machine Learning)*. A Bradford Book, England, 1998.
- [2] Neil Smyth. *Android Studio Development Essentials : Android 5 Edition*. CreateSpace, England, 2014.
- [3] Gerald Tesauro. Temporal Difference Learning and TD-Gammon. *Communications of the ACM*, 38(3), mar 1995.