Puissance 4

- Présentation générale du projet :

Le but de ce projet était de développer un jeu de puissance 4 jouable contre une IA ou en joueur contre joueur, que ce soit en local ou en réseau. Le jeu du puissance 4 se joue à deux et consiste en une grille de sept colonnes par six lignes et de deux types de pions : un par joueur. Le but du jeu est alors d'aligner quatre de ses pions en ligne, colonne ou diagonale. Nous avons choisi ce projet car l'Intelligence Artificielle est une notion très importante de nos jours, et ce projet nous a permi de s'y intéresser de plus près. De plus, la plupart des jeux vidéo possèdent un mode en ligne, d'où l'intérêt de cet aspect du projet.

- Présentation des deux parties principales du programme :

1. Partie Intelligence Artificielle

Une intelligence artificielle est un programme qui permet de simuler une réflexion humaine. Ce type de programme est notamment utilisé en robotique ou en automatique, dans les jeux vidéo, etc...

En premier lieu, il est important de préciser que le jeu du puissance 4 est un jeu dans lequel chaque joueur peut savoir à n'importe quel moment l'état de la partie, ses possibilités de jeu et les possibilités de jeu de son adversaire. Cette situation est alors propice à l'utilisation d'un algorithme MinMax.
Le principe de cet algorithme est très simple : l'ordinateur teste toutes ses possibilités de jeu, puis toutes les possibilités de jeu de l'adversaire, etc.. L'algorithme sélectionne ensuite le coup le plus favorable, c'est à dire le coup qui lui ouvrira le plus de possibilités de victoire. On peut alors modéliser son fonctionnement sous la forme d'un arbre parcouru jusqu'en bas (situation de gain ou de perte), puis remonté. L'algorithme renvoie alors le coup le plus favorable à jouer. Le nom "MinMax" vient du fait que l'algorithme cherche à MINimiser la perte MAXimum. Dans tout le processus, on suppose que le joueur humain joue au mieux, c'est à dire qu'il ne fait pas d'erreur.

Dans l'idéal, cet algorithme teste donc toutes les possibilités de jeu, mais cela devient impossible si le jeu est trop complexe : le temps de calcul sera alors trop long. C'est le cas du puissance 4 et c'est pour cela qu'il faut mettre en place un système de profondeur. L'algorithme ne va alors plus tester toutes les possibilités, mais seulement un certain nombre de coups.
Cela nécessite une fonction d'évaluation qui va pondérer chaque situation afin de choisir la plus favorable à n'importe quel instant. Pour se faire, nous avons choisi la méthode suivante : au moment d'évaluer, l'algorithme compte le nombre maximum de pions alignés avec possibilité de gain, c'est à dire le nombre maximum de pions alignés avec des cases vides à côté pour espérer y mettre des pions et gagner.

Dans un soucis d'optimisation du programme, nous avons aussi intégré l'élagage AlphaBeta dans notre programme. Cet algorithme permet de limiter l'exploration de l'arbre en coupant

la recherche lorsque l'on sait que l'on ne trouvera pas de score plus favorable dans la branche qui arrive.

2. Partie jeu en réseau

Aujourd'hui, la majorité des jeux vidéo intègrent, en plus d'un mode local, une possibilité de jeu en ligne. Ceci est particulièrement intéressant lorsque le jeu fait se rencontrer deux joueurs, en face à face. Nous avons donc décidé d'intégrer un mode "jeu en ligne" à notre Puissance 4.

Pour rendre cela possible, nous avons choisi de passer par une base de donnée NoSQL en temps réel, disponible gratuitement *via* Firebase[1], qui est un service de Google. On va alors pouvoir mettre deux terminaux en relation avec cette base de données, organisée en plusieurs salons de jeu (afin de permettre plusieurs parties en simultané). Une fois 2 joueurs connectés au même salon, la partie peut alors commencer. Cette dernière sera répartie entre la machine locale et la base de données en ligne. Le programme Python est donc subdivisé en deux fichiers, le premier prenant en charge les fonctionnalités de base du jeu, le second prenant en charge la connexion à la base de données, l'envoi et la récupération des valeurs. Le programme va donc fonctionner de la même manière qu'en 1 contre 1 local pour tout ce qui concerne l'affichage et la gestion de la partie (sortie de jeu, fin de partie, etc…). La seule différence se trouve essentiellement au moment de jouer un coup. À ce moment là, soit le joueur concerné est celui qui joue le coup – dans ce cas son coup s'affiche sur sa machine puis est envoyé au serveur – soit il est celui qui ne joue pas – dans ce cas le jeu va uniquement récupérer la valeur au niveau de la base de données, puis afficher le coup. Après cela, chaque machine va vérifier si un joueur a gagné. Si un joueur a gagné, chaque machine va demander au joueur s'il veut rejouer, puis va envoyer sa réponse sur la base de données. La partie recommence alors si les deux joueurs ont répondu "oui". De plus, comme dans une partie locale, chaque joueur peut rentrer son nom. Le nom que le joueur aura rentré sera donc écrit sur la base de données, et le programme lira le nom de son adversaire dès qu'il aura rejoint le salon.

- Résultats obtenus, améliorations possibles :

Le jeu, aussi bien en local qu'en ligne, est entièrement fonctionnel. Cependant, certains points pourraient être améliorés, notamment quelques bugs occasionnels. De plus, en retravaillant la fonction d'évaluation, il serait possible de rendre l'IA réellement imbattable pour des profondeurs relativement faibles (et donc envisageables sur un PC classique). Il serait aussi intéressant de faire jouer l'IA aléatoirement parmis les colonnes les plus favorables (actuellement, la colonne sélectionnée est la première explorée dans l'arbre, c'est à dire celle la plus à droite).

[1] https://firebase.google.com/products/realtime-database/