|  |  |  |
| --- | --- | --- |
|  |  | |
| Rapport AI GAME | |  |
|  |  | |
|  |  | |
|  | Yohann TOGNETTIYann MARTIN D’ESCRIENNE | |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | DESCRIPTION DE NOTRE IA**Fonction d’évaluation :**  La fonction d’évaluation de notre IA est la fonction d’évaluation naîve, c-à-d :    De plus, même lorsque l’IA trouve un **coup gagnant**, on cherche alors à maximser l’écart de points.  Nous avons essayer un fonction d’évaluation plus complexe mais aucun gain significatif s’est fait ressentir, l’IA avait d’ailleurs tendance à perdre plus souvent.  **Profondeur Dynamique :**  Notre IA possède un **profondeur de base de 9**. Mais un système de profondeur dynamique à été mis en place. En effet lorsque le coup précédent de l’IA à pris moins de **450 ms, la profondeur augmente** alors de 1 (de 2 si moins de 10ms). De même lorsque l’IA prend trop de temps pour réfléchier (env 1,3) **la profondeur diminue alors de 1** pour ne pas trop dépasser la règle des deux secondes.  Elle possède également une **profondeur minimale lorsque le tableau de jeu se réduit** car à ce moment, le temps de calcul réduit généralement drastiquement. A ce moment, on atteind généralement la profondeur maximale (22)  **Multi-Thread :**  Notre IA utiliser un multi-thread basique en créant un thread par position de jeu de départ, soit **12 thread au début** puis **6 quand le tableau est réduit**. Le thread vont alors faire un **min-max** classique avec des **coupes Alpha-Beta.** Les threads vont alors écrire leur résultat dans un tableau du thread principal et le min ou le max sera choisi selon le joueur.  **Entrée de jeu :**  Suite à des tests avec des grandes profondeur, il s’est avéré que commencer avec un coup sur la position 6 en joueur P1 était le plus rentable. De même, en joueur P2, certains coups sont plus avantageux. Il suivent le calcul si dessous :   * Si le joueur 1 joue P tel que P%4 < 2 alors P2 doit jouer P’ = (P +9) %24 * Si le joueur 1 joue P tel que P%4 >= 2 alors P2 doit jouer P’ = (P+19) %24   **Optimisation de compilation :**  Nous avons utiliser une compilation optimisé avec d’obtenir un temps de réponse plus rapide. Voici la commande gcc :  g++ -fno-stack-protector -fno-pie -no-pie -fno-pic -O3 -static -Wall \*.cpp -o program | |  |
|  |  |