Compte Rendu du Projet : Jeu d'Hex avec MCTS en C++

Étudiants: *Dorian Biagi, Giovanni Carré, Loïc Berthelot*

Introduction

Ce compte rendu présente un projet réalisé en L3, portant sur le développement du jeu d'Hex avec deux joueurs basés sur des algorithmes de Monte Carlo Tree Search (MCTS). L'objectif du projet était de créer un solveur de jeu d'Hex performant, capable de rivaliser avec différents types de joueurs, notamment aléatoires et humains, tout en optimisant le code pour minimiser le temps de calcul.

L'algorithme de **Monte-Carlo** pour le jeu d'hex est une simulation probabiliste utilisée pour évaluer les positions dans le jeu. Il fonctionne en effectuant des simulations aléatoires de parties à partir d'une position donnée, puis en utilisant les résultats de ces simulations pour estimer la qualité de la position. Plus on effectue de simulations, plus l'estimation devient précise. Cela permet aux joueurs de prendre des décisions éclairées sur les coups à jouer en fonction des probabilités de succès de chaque coup.

Description du Projet

Fonctionnalités Principales

Le projet consistait à développer un jeu d'Hex basé sur l'algorithme MCTS, avec deux joueurs IA. Les points clés du projet étaient les suivants :

1. ***Tournoi MCTS:*** *La promotion est chargée de réaliser un tournoi avec 10 équipes, afin de déterminer quelle équipe avait créé le meilleur joueur IA basé sur l'algorithme MCTS.*
2. ***Contraintes de temps:*** *Chaque joueur IA a une limite de temps très stricte pour prendre chaque coup, généralement de l'ordre de quelques millisecondes. Si un joueur dépasse cette limite, il perd la partie.*
3. ***Optimisation de la Vitesse:*** *Une grande partie du projet consiste à optimiser le code pour maximiser la vitesse de calcul des joueurs IA. Cela a impliqué l'optimisation des algorithmes MCTS eux-mêmes, ainsi que l'optimisation du code source fourni par l'université d'Angers.*
4. ***Stratégie Hors MCTS:*** *En cas de situation dans laquelle le jeu ne se trouve pas dans l'arbre de recherche MCTS, une stratégie de coût est élaborée en utilisant un algorithme de recherche du meilleur coup basé sur des chemins possibles.*

Modes de Jeu

Le projet comporte deux modes de jeu principaux :

1. ***Mode Apprentissage:*** *Dans ce mode, le jeu doit être capable d'apprendre rapidement, car le jeu Hex offre de nombreuses possibilités. À chaque palier, un fichier* `intelligence.txt` *est créé pour enregistrer les différents arbres de recherche MCTS. Afin d'optimiser l'utilisation de l'espace de stockage, une technique d'encodage est utilisée pour représenter les coups de manière compacte.*
2. ***Mode Jouable:*** *Dans ce mode, le jeu doit être capable de charger et parcourir l'arbre MCTS. Si l'état actuel du jeu n'est pas présent dans l'arbre MCTS, le jeu doit être capable de calculer le meilleur coup possible en moins de 7 millisecondes.*

Méthodologie

Le projet a été réalisé en suivant les étapes suivantes :

1. ***Conception:*** *Une conception détaillée a été élaborée, décrivant les algorithmes MCTS, les structures de données et les mécanismes de jeu.*
2. ***Implémentation:*** *Le jeu a été implémenté en utilisant les langages de programmation appropriés, en optimisant le code pour respecter les contraintes de temps.*
3. ***Tests:*** *Le jeu a été testé contre différents types de joueurs, y compris des joueurs aléatoires, des robots et des joueurs humains. Des ajustements ont été apportés aux stratégies en fonction des performances observées.*
4. ***Tournoi:*** *Le tournoi avec les 10 équipes a été organisé pour évaluer les performances de l'IA MCTS.*

Résultats

Les résultats du projet ont montré que l'équipe avait réussi à créer un joueur IA MCTS performant, capable de rivaliser avec succès contre d'autres équipes, arrivant en première place du classement. L'optimisation du code a permis de respecter les contraintes de temps strictes, et les stratégies de jeu ont été élaborées avec succès pour les situations en dehors de l'arbre de recherche MCTS.

Conclusion

En conclusion, le projet de développement du jeu d'Hex avec MCTS est un succès, les objectifs principaux ont été atteints. L'équipe a pu créer un joueur IA MCTS compétitif. Ce projet a été une expérience enrichissante en matière de conception, d'optimisation et de développement de jeux basés sur des algorithmes d'intelligence artificielle.

Pour plus de détails, sur le code, regardez le ReadMe dans le code source, le code est aussi bien entendu commenté.
Pour lancer le jeu en mode learning ou non : mettre dans le “main” la variable learning à true ou false.

Pour exécuter le projet :

Linux :

***Installer CMake** : La plupart des distributions Linux disposent de CMake dans leurs gestionnaires de paquets. Vous pouvez installer CMake avec la commande appropriée pour votre distribution. Par exemple, sur Ubuntu, utilisez :*
bash
```
sudo apt-get install cmake
```

1. ***Installer un compilateur C++** : Vous aurez besoin d'un compilateur C++ pour compiler votre projet. La plupart des distributions Linux incluent GCC (GNU Compiler Collection) par défaut. Vous pouvez vérifier si GCC est installé en exécutant :*
   bash

```
g++ --version
```

2. *Si GCC n'est pas installé, vous pouvez l'installer à partir de votre gestionnaire de paquets.*
3. ***Créer un dossier pour votre projet** : Créez un dossier pour votre projet et placez-y votre fichier `CMakeLists.txt`.*
4. ***Ouvrir un terminal** : Ouvrez un terminal de ligne de commande.*
5. ***Naviguer vers le dossier du projet** : Utilisez la commande `cd` pour vous déplacer vers le dossier de votre projet. Par exemple :*
   bash

```
cd /chemin/vers/votre/projet
```

6. ***Créer un dossier pour la construction** : Il est recommandé de créer un dossier séparé pour la construction de votre projet. Exécutez la commande suivante pour créer un dossier "build" dans votre projet :*
   bash

```
mkdir build
```

```
cd build
```

7. ***Générer les fichiers de construction avec CMake** : Exécutez la commande `cmake` avec le chemin vers votre fichier `CMakeLists.txt` depuis le dossier de construction. Par exemple :*
   bash

```
cmake ..
```

8. *La notation `..` signifie que vous faites référence au répertoire parent (où se trouve votre `CMakeLists.txt`).*
9. ***Compiler avec `make`** : Après avoir généré les fichiers de construction avec succès, vous pouvez utiliser `make` pour compiler votre projet. Exécutez simplement la commande `make` depuis le dossier de construction :*
   bash

```
make
```

10. ***Exécuter votre programme** : Une fois la compilation terminée avec succès, vous pouvez exécuter votre programme depuis le dossier de construction.*

Sur windows :

1. **Installer CMake** : Téléchargez et installez CMake à partir du site officiel de CMake pour Windows : https://cmake.org/download/
2. **Installer un compilateur C++** : Pour utiliser `make`, vous aurez besoin d'un compilateur C++ tel que MinGW ou Microsoft Visual C++. Vous pouvez télécharger MinGW à partir de https://mingw-w64.org/doku.php ou installer Visual Studio avec le support de développement C++.
3. **Créer un dossier pour votre projet** : Créez un dossier pour votre projet et placez-y votre fichier `CMakeLists.txt`.
4. **Ouvrir un terminal** : Ouvrez un terminal de ligne de commande, par exemple, l'invite de commande Windows ou PowerShell.
5. **Naviguer vers le dossier du projet** : Utilisez la commande `cd` pour vous déplacer vers le dossier de votre projet. Par exemple :
   bash

```
cd Chemin\vers\votre\projet
```

6. **Créer un dossier pour la construction** : Il est recommandé de créer un dossier séparé pour la construction de votre projet. Exécutez la commande suivante pour créer un dossier "build" dans votre projet :
   bash

```
mkdir build
cd build
```

7.
8. **Générer les fichiers de construction avec CMake** : Exécutez la commande `cmake` avec le chemin vers votre fichier `CMakeLists.txt` depuis le dossier de construction. Par exemple :
   bash

```
cmake chemin\vers\votre\projet
```

9. Assurez-vous que CMake se trouve dans votre chemin d'exécution, sinon, vous devrez spécifier le chemin complet vers CMake.

10. **Compiler avec `make`** : Après avoir généré les fichiers de construction avec succès, vous pouvez utiliser `make` pour compiler votre projet. Exécutez simplement la commande `make` depuis le dossier de construction :
    bash

```
make
```

11. Si vous utilisez Visual Studio, vous pouvez exécuter `msbuild` au lieu de `make` pour la compilation.
12. **Exécuter votre programme** : Une fois la compilation terminée avec succès, vous pouvez exécuter votre programme depuis le dossier de construction.