Hugo Malglaive **04/05/2021**

USSI0R : Apprentissage et Intelligence artificielle

Compte Rendu du projet

Rappel du sujet : Projet 3 Les algorithmes de jeux

Nous disposons d'un espace d'états modélisant les actions de deux joueurs (MAX et MIN). Les feuilles correspondent aux états terminaux du jeu. La racine est un nœud MAX. Les valeurs des états terminaux sont indiquées en bas de chaque état. Les entrées de l'algorithme sont le graphe (sommets, transitions et les valeurs des feuilles) et le sens d'exploration (de la gauche vers la droite ou l'inverse). Le résultat de l'algorithme est le graphe (les états explorés par l'algorithme alpha-beta, les transitions correspondantes, à côté de chaque état exploré la valeur correspondante à la terminaison de l'algorithme et les branches coupées).

Lien git du code source : https://github.com/hugoTortuga/AI_AlphaBeta

Introduction

J'ai choisi ce sujet car il me correspond, je suis intéressé par les algorithmes de résolution de jeux comme les échecs. J'ai commencé par développer l'algorithme Mini_Max car il est plus simple que l'algorithme Alpha_Beta . Pour ce sujet, j'ai beaucoup utilisé la récursivité, car on manipule ici des graphes, des arbres.

Entrées du programme

```xml
<?xml version="1.0" encoding="utf-8"?>
<tree>
    <node>
      <node>
        <node>
          <node value="-1"/>
          <node value="3"/>
        </node>
        <node>
          <node value="5"/>
          <node value="1"/>
        </node>
      </node>
      <node>
        <node>
          <node value="-6"/>
          <node value="9"/>
        </node>
        <node>
          <node value="0"/>
          <node value="-9"/>
          <node value="10"/>
        </node>
      </node>
    </node>
</tree>
```

Les entrées de mon algorithme sont : un arbre, avec une racine (root), des enfants. Pour alimenter le programme, on donne des fichiers XML en paramètres qui ont cette forme :

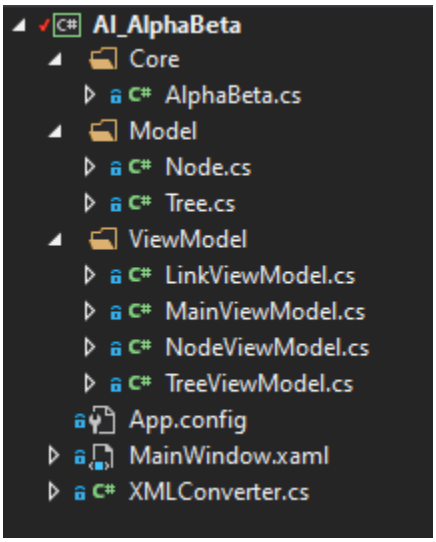
Le fichier XML comporte une balise <tree> puis une racine <node>, ensuite, cette racine, a des enfants <node> qui ont des enfants <node>, jusqu'aux feuilles, qui ont, elles, une valeur.

Le choix du langage C# : Pourquoi ?

Le C# est le langage avec lequel je suis le plus à l'aise. De plus, je savais que pour l'affichage graphique Windows Presentation Form (WPF) serait très pratique. Mon application utilise le pattern Model – View – ViewModel (MVVM), très similaire au pattern Model View Controller.

Le code a été entièrement documenté pour une meilleure compréhension.

L'architecture de mon programme :



Mon programme comporte un package Core, qui est le cœur de l'algorithme AlphaBeta, un package Model, un package ViewModel, un fichier MainWindow.xaml (la vue de mon application en WPF) et enfin une classe utilitaire XMLConverter qui transforme ma data XML en *AI_AlphaBeta.Model.Tree*

Mon Model :

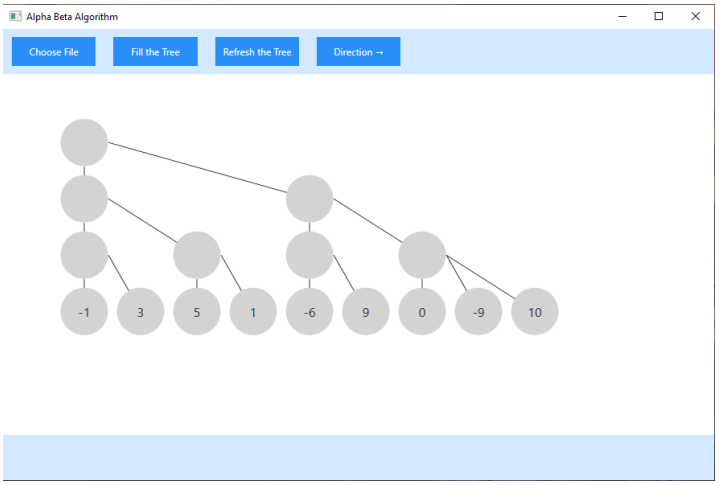
Mon package model est très simple, il comporte une classe Tree, qui a un nœud racine, et une classe nœud, qui a une valeur numéraire, un booléen indiquant s'il a été visité ou non et une liste de nœuds enfants.

Librairie Graphique : Telerik

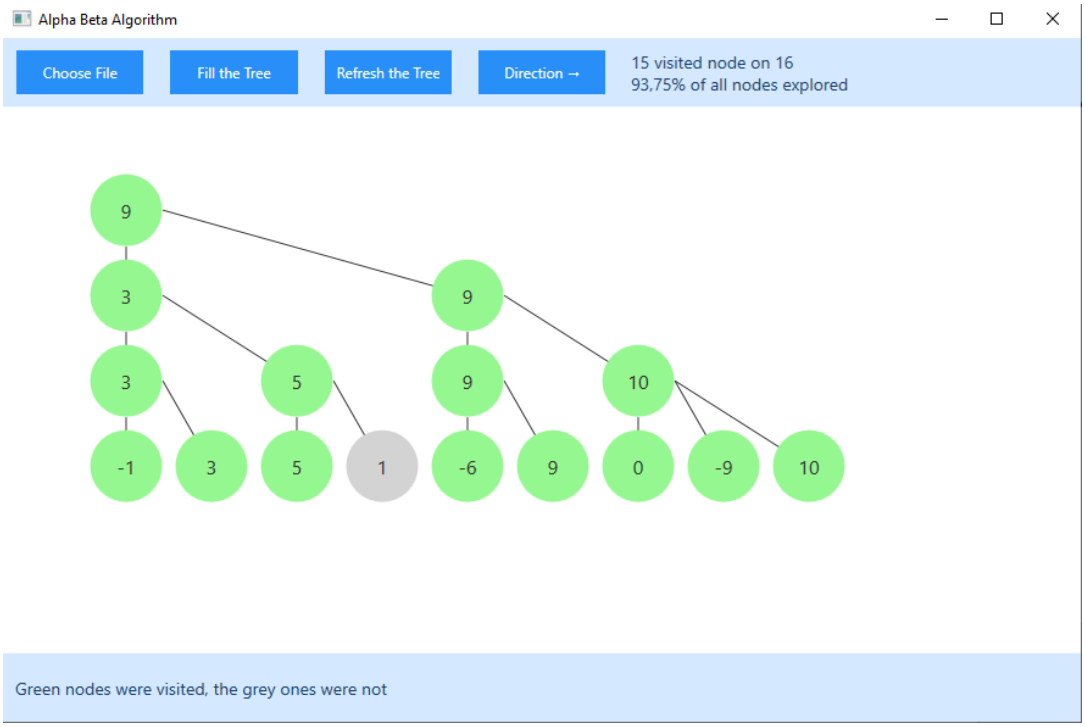
J'ai choisi d'utiliser la bibliothèque graphique Telerik, car elle propose un composant appelé RadDiagram qui permet d'afficher des graphes, des nœuds et des liens entre les nœuds. Pour permettre à mon arbre d'être affiché, j'ai dû implémenter la structure IGraphSource fournit par Telerik. Cette interface comporte une liste de nœuds et une liste de liens entre les nœuds. J'ai donc un petit morceau de code qui permet de transformer ma structure Tree, vers mon implémentation d'IGraphSource.

L'application :

Dans un premier temps, vous devez sélectionner un fichier XML qui a une structure correcte. L'arbre s'affiche ensuite, On peut zoomer et dézoomer l'arbre, notamment pour les gros arbres.



Ensuite, on peut remplir l'arbre :



On peut également rafraichir, l'arbre, ou annuler l'exploration, ou encore changer la direction.

Attention : Pour **changer la direction** , vous devez d'abord rafraichir l'arbre, puis le remplir à nouveau, sans quoi, rien ne se passe.

Remarque : En fonction de la direction d'exploration choisi, l'arbre change de forme. Si on change de direction pour l'arbre ci-dessous, on obtient :