Лабораторна робота №2

З предмету “Програмування інтелектуальних інформаційних систем”

Студента групи ІП-01

Черпака Андрія

Тема: Алгоритми мінімакс та альфа-бета відсікання

Київ-2022

Абстрактний клас ***DecisionTree***, від якого наслідуються ***AlphaBetaAlgo*** та ***MiniMaxAlgo***

```
using System;

namespace Lab2.Model;

public abstract partial class DecisionTree
{
    public Node Root;
    private static int _depthLvlCtr;
    internal Game game;

    public DecisionTree(Position startState, short firstI, short firstJ,
int currentMaxDepth, Game gm)
    {
        game = gm;
        if (startState.DistancesToEnemies.Length > 1)
            throw new NotImplementedException("This application can't
work with more than 1 enemy for now :(");
        _depthLvlCtr = currentMaxDepth;
        Root = new Node(startState, firstI, firstJ, 0, gm);
    }

    protected abstract int GetNextMoveNodeValue(Node node, int depth);

    public bool ChooseNextMove(out short i, out short j, int depth)
    {
        int bestVal = GetNextMoveNodeValue(Root, depth);
        bool found = false;
        i = j = -1;
        foreach (var node in Root.PossibleNextMoves)
        {
            if (node.Value == bestVal && (!found || node.IsTerminal))
            {
                i = node.I;
                j = node.J;
                found = true;
                if (node.IsTerminal) return true;
            }
        }

        return found;
    }

    public void MoveIsDone(int i, int j)
    {
        foreach (Node n in Root.PossibleNextMoves)
        {
            if (n.I == i && n.J == j)
            {
                Root = n;
                _depthLvlCtr++;
                Root.InitializeNewNodes();
                GC.Collect();
                return;
            }
        }
```

```
            throw new IndexOutOfRangeException();
        }

        public bool RootIsTerminal() => Root.IsTerminal;
}
```

Клас ***MiniMaxAlgo***

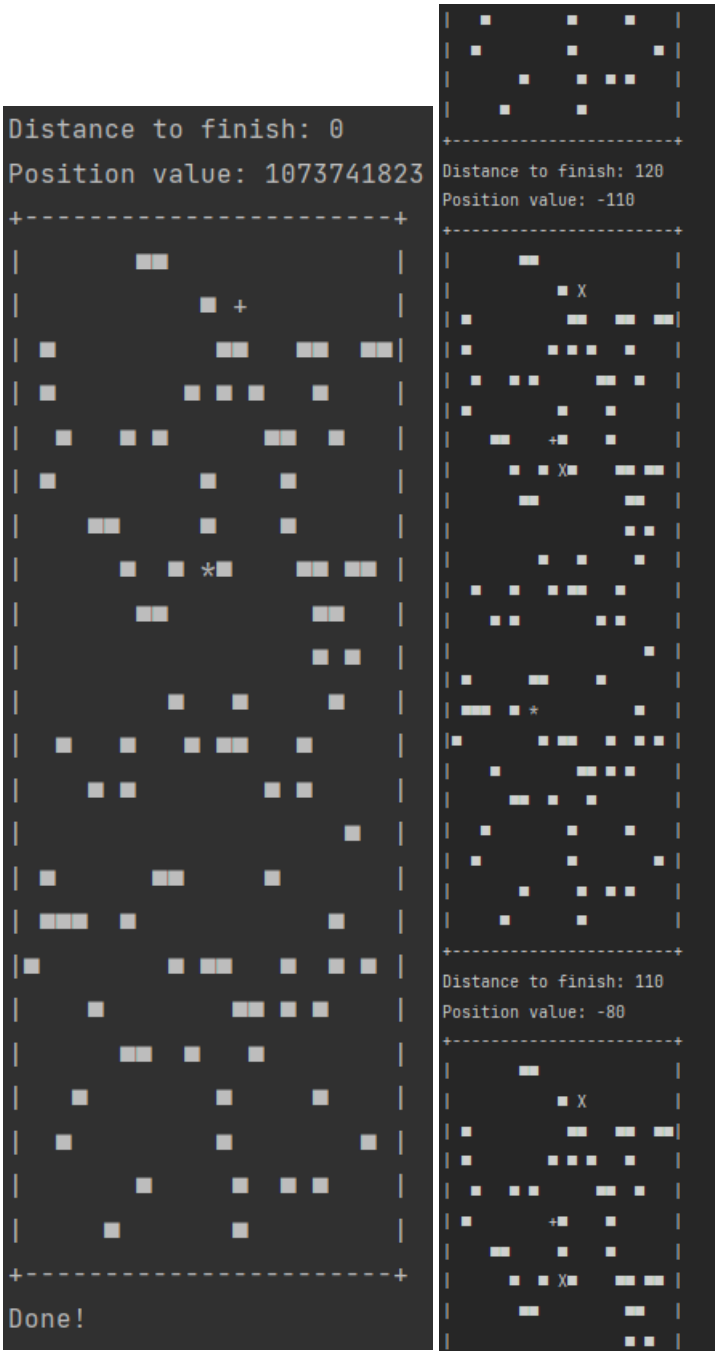
```
namespace Lab2.Model;

public class MiniMaxAlgo : DecisionTree
{
    public MiniMaxAlgo(Position ss, short i, short j, int md, Game g) : base(ss,
i, j, md, g) {}
    protected override int GetNextMoveNodeValue(Node node, int depth)
    {
        return MiniMax(node, depth);
    }

    private int MiniMax(Node node, int depth)
    {
        if (depth == 0 || node.IsTerminal)
        {
            node.Value = (int)node.CurrentState.Value();
        }
        else
        {
            bool maximize = node.DepthLevel % 2 == 0;
            node.Value = maximize ? Int32.MinValue / 2 : Int32.MaxValue / 2;
            foreach (Node child in node.PossibleNextMoves)
            {
                int newValue = MiniMax(child, depth - 1);
                if (maximize && newValue > node.Value || !maximize && newValue <
node.Value) node.Value = newValue;
            }
        }

        return node.Value;
    }
}
```

Клас ***AlphaBetaAlgo***

```
namespace Lab2.Model;

public class AlphaBetaAlgo : DecisionTree
{
    public AlphaBetaAlgo(Position ss, short i, short j, int md, Game g) :
base(ss, i, j, md, g) {}
    protected override int GetNextMoveNodeValue(Node node, int depth)
    {
        return AlphaBeta(node, depth);
    }

    private int AlphaBeta(Node node, int depth, int α = Int32.MinValue/2, int β
= Int32.MaxValue/2)
    {
```

```
        if (depth == 0 || node.IsTerminal)
        {
            node.Value = (int)node.CurrentState.Value();
        }
        else if (node.DepthLevel % 2 == 0)
        {
            node.Value = Int32.MinValue/2;
            foreach (Node child in node.PossibleNextMoves)
            {
                int newValue = AlphaBeta(child, depth - 1, α, β);
                if (newValue > node.Value) node.Value = newValue;
                if (α < node.Value) α = node.Value;
                if (node.Value >= β) break;
            }
        }
        else
        {
            node.Value = Int32.MaxValue/2;
            foreach (Node child in node.PossibleNextMoves)
            {
                int newValue = AlphaBeta(child, depth - 1, α, β);
                if (node.Value > newValue) node.Value = newValue;
                if (β > node.Value) β = node.Value;
                if (node.Value <= α) break;
            }
        }

        return node.Value;
    }
}
```

Результати роботи обох алгоритмів у мене збігаються, хоча час роботи, звісно ж, відрізнятиметься. В консолі покроково змінюватиметься стан поля, а також на кожному кроці відображатиметься оцінка вузла та відстань до фінішу. Також хід гри записується до файлу.


Distance to finish: 0
Position value: 1073741823
Done!
Distance to finish: 120
Position value: -110
Distance to finish: 110
Position value: -80