# **1 Анализ технических требований**

## **Анализ задания и выбор технологии, языка и среды разработки**

В данной курсовой работе рассматривается тема компьютерной визуализации методов теории игр и принятие решений.

Теория игр является разделом дисциплины «Исследование операций», который занимается изучением оптимальных стратегий в играх, в основном антагонистических, когда существуют несколько соперничающих сторон. Поэтому целью данной курсовой работы является разработка программы, которая находила бы оптимальную стратегию, приводящую одну из сторон к победе в какой-либо игре.

Для реализации данной программы необходимо создание множества объектов, которые будут представлять положение игроков на поле, варианты ходов, эвристические оценки. Для этого была выбрана среда разработки Visual Studio и язык программирования C#, так как он поддерживает объектно-ориентированный подход разработки, что существенно облегчит взаимодействие между объектами. Визуальное составляющее данного приложения будет создано с помощью системы для построения клиентских приложений WPF (Windows Presentation Foundation).

**1.2 Анализ процесса обработки информации и построение функциональных диаграмм**

Данная программа является продуктом взаимодействия нескольких сущностей или актеров, которыми могут являться как программные компоненты, так и непосредственно пользователи (см. рисунок 1).

На данной диаграмме показано взаимодействие пользователя с компьютером.

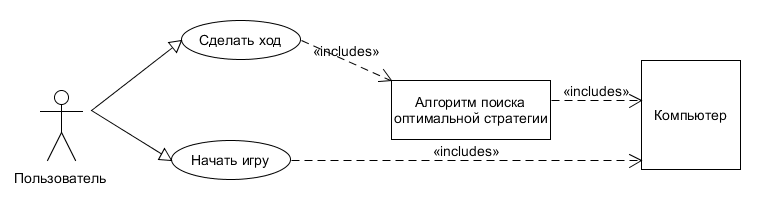
****

Рисунок 1 – диаграмма последовательности действий

Также для получения более полного представления о работе программы, ориентируясь на последовательность совершения различных операций, была реализована диаграмма последовательности (см. рисунок 2).

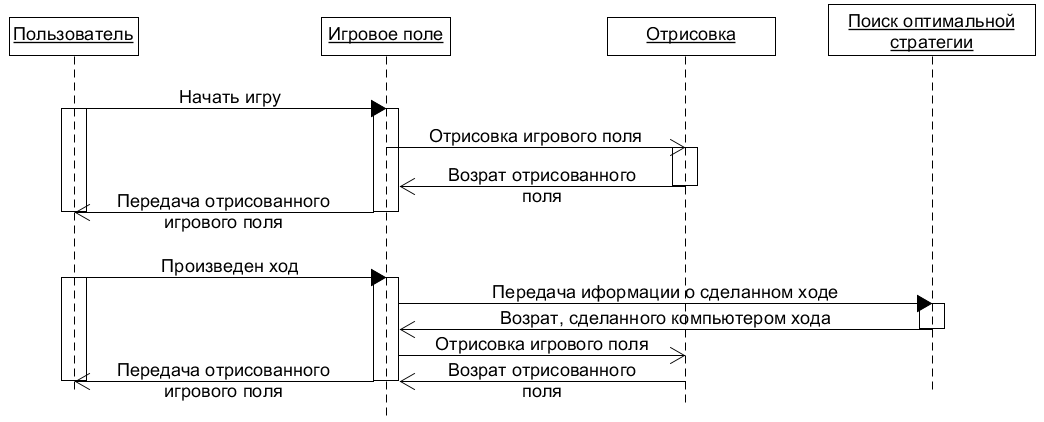


Рисунок 2 – диаграмма последовательности

**1.3 Анализ хранимой информации и выбор структур данных для ее представления**

В данной курсовой работе будут использоваться различные структуры данных:

1. Динамические массивы
2. Очереди
3. Классы
4. Элементы отрисовки
5. Пользовательские элементы управления
6. Древовидные структуры

Данная программа не будет производить запись и хранение данных.

## **1.4 Выбор методов и алгоритмов решения задачи**

Для нахождения оптимальной стратегии будет использоваться алгоритм «Минимакс» и алгоритм его оптимизации «Альфа-бета отсечение».

«Минимакс» используется в играх с нулевой суммой, то есть там, где выигрыши двух противоборствующих сторон противоположны. Его главной цель является выбор хода, дающего противнику наихудший набор возможностей.

Алгоритм используется в основном в традиционных настольных играх, где:

1. Информация полная;
2. Обстановка статична;
3. Пространство действия дискретно;
4. Действия предопределены.

Антагонистическая игра может быть представлена тройкой <X, Y, F>, где X и Y — множества стратегий первого и второго игроков, соответственно; F — функция выигрыша первого игрока, ставящая в соответствие каждой паре стратегий (ситуации) (x,y) [1]

Данный алгоритм рассматривает двух игроков:

1. Игрок Min, выбирающий ход с наименьшим значением;
2. Игрок Max, выбирающий ход с наибольшим значением.

На каждый ход строится дерево игры, где содержатся все ходы, которые могут быть сделаны каждым из игроков. Каждый ход имеет свою оценку, например, 5 – победа, -5 – проигрыш, 2 – ничья (см. рисунок 3). Глубина раскрытия дерева может определяться игроком самостоятельно. Чем глубже происходит раскрытие, тем более оптимально совершаются ходы. Это своего рода уровень сложности.

Рассмотрим рисунок 1 подробнее. Игроку Max необходимо решить какой ход более оптимальный, с помощью алгоритма «Минимакс» строится дерево игры. У него есть 3 варианта хода.

Раскроем левую ветку. Если он поставит «крестик» в нижний левый угол, то у игрока Min останется два варианта хода и как видно самым нежелательным исходом для игрока Max при реализации этой ветки может стать ничья, а самым положительным победа. Поэтому можно подумать, что игрок Max будет идти этим путем, так как там гарантирована победа, но так как игрок Min также ходит оптимально, то он скорее согласится на ничью, нежели на победу Max, следовательно, он выбирает минимальное значение (2 – ничья). А значит игрок Max не выиграет, но и не проиграет.

Раскрывая среднюю и правую ветку наихудшим результатом для Max станет проигрыш, и игрок Min это знает, поэтому даже, если в этих ветках есть выигрышные комбинации, игрок Min никогда не даст их достичь. Следовательно, средняя и правая ветка на втором уровне вложенности получают минимальную оценку -5 (проигрыш).

Таким образом игрок Max может рассчитывать максимум на ничью.

В этом и заключается алгоритм «Минимакс». Если же в игре нет возможности просчитать все возможные комбинации до конца игры, то каждой игровой единице (фигуре) присваивается числовое значение (очки значимости) от меньшего к большему, и чем больше игрок Max получит очков при каком-либо ходе, тем меньше получит игрок Min и наоборот.

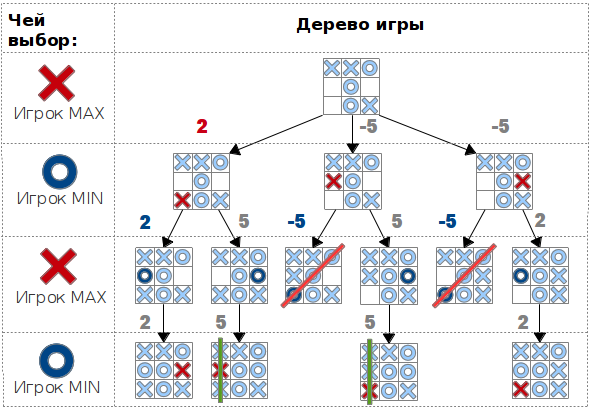


Рисунок 3 – дерево игры

Так как при работе данного алгоритма количество комбинаций может достигать больших значений (см. таблицу 1), то необходимо производить оптимизацию при построении дерева решений. В этом помогает алгоритм оптимизации «Альфа-бета отсечение», основной принцип которого: “Если у вас есть плохая идея, не нужно тратить время на то, чтобы выяснить, насколько она плоха”.

Таблица 1 – уровни сложности игр

|  |  |  |  |
| --- | --- | --- | --- |
| Игра | Размер доски | Коэффициент ветвления | Количество листьев |
| Крестики-нолики | 3x3 | 4 | 10^5 |
| Четыре в ряд | 6x7 | 4 | 10^21 |
| Шашки | 8x8 | 2.8 | 10^31 |
| Шахматы | 8x8 | 35 | 10^123 |
| Го | 19x19 | 250 | 10^360 |

Рассмотрим алгоритм оптимизации «Альфа-бета отсечение» на примере рисунка 4.

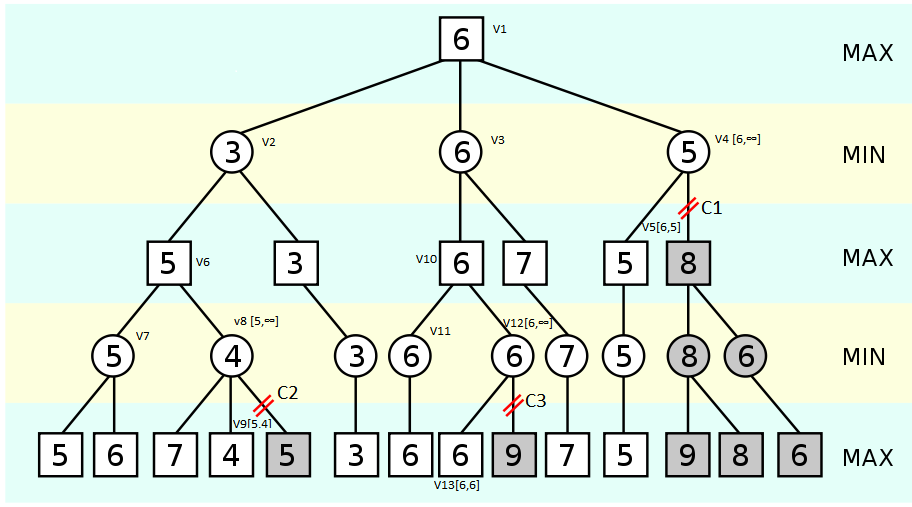


Рисунок 4 – альфа-бета отсечение

Здесь опять используется алгоритм «Минимакс», игрок Max имеет 3 варианта хода, ему необходимо совершить наиболее оптимальный.

В данном алгоритме вводится понятия альфы и беты, причем если раскрытие происходит за игрока Max и оценка оказывается больше беты, или раскрытие происходит за игрока Min и оценка оказывается меньше альфа, то дальнейшее рассмотрение вершин становится не целесообразным. [1]

Рассмотрим первое отсечение (C1). После того, как была полностью пройдена ветка V3, где f(V3) = 6, то альфа стала равна 6, далее альфа передается в узел V4, где после раскрытия узла V5 бета = 5, получается, что бета>альфа, то дальнейшее рассмотрение этой ветки становится нецелесообразно, так как игрок Max не выберет ход, где получит 5 очков вместо хода V3, где получит 6 очков. И независимо от того какое число будет в отсечении C1 игроку Max не выгодно ходить данной комбинацией.

Рассмотрим второе отсечение (C2). При раскрытии ветки V8 мы получили, что альфа = 4 (минимальное число), а затем в отсечении C2 мы обнаружили ход со значимостью 5, а так как в этом случае бета>альфа, то раскрытие этой ветки становится нецелесообразно, так как игрок Min не выберет ход, где игрок Max получит 5 очков вместо 4 очков.

Третье отсечение (C3) производится по тому же алгоритму, что и второе отсечение (C2).

Источники

1 https://www.dokwork.ru/2012/11/tictactoe.html