

**МИНОБРНАУКИ РОССИИ**

**федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технологический университет «СТАНКИН»**

**(ФГБОУ ВО «МГТУ «СТАНКИН»)**

Институт цифровых интеллектуальных систем  
Кафедра компьютерных систем управления

Образовательная программа 15.03.04

«Автоматизация технологических процессов и производств»

# Дисциплина «Операционные системы и базы данных»

**Отчет**

**по лабораторной работе №1**

Тема: «Уголки углом. Добавление компьютерного игрока. Реализация простого искусственного интеллекта»

Выполнил:

студент гр. АДБ-19-08 29.05.2021 Поздняков Александр

(дата) (подпись)

Принял:

Доцент, к.т.н. \_\_\_\_\_\_\_\_\_\_\_\_\_ Пушков Р.Л.

(дата) (подпись)

**Москва 2021**

На основе проекта из лабораторной ПрПр 4, продолжим работу. Переименуем класс TicTacPlayer на TicTacHumanPlayer, а также создадим TicTacPlayer, для которого предыдущий класс будет дочерним.

Код заголовка класса:

#pragma once

#include "pch.h"

#include "TicTacBoard.h"

class TicTacPlayer

{

protected:

TicTacBoard\* board;

CellType cellType;

string name;

public:

void SetupPlayer(string name, CellType cellType);

void SetBoard(TicTacBoard\* board);

virtual bool MakeMove() = 0;

string GetName();

};

Реализация класса:

#include "TicTacPlayer.h"

void TicTacPlayer::SetupPlayer(string name, CellType cellType)

{

this->name = name;

this->cellType = cellType;

}

void TicTacPlayer::SetBoard(TicTacBoard\* board)

{

this->board = board;

}

string TicTacPlayer::GetName()

{

return this->name;

}

Модифицирован класс TicTacPlayer, из него выброшены общие функции, данный класс унаследовали от TicTacPlayer, а также опреден метод MakeMove.

Изменения в коде:

#pragma once

#include "TicTacPlayer.h"

class TicTacHumanPlayer : public TicTacPlayer

{

public:

TicTacHumanPlayer();

virtual ~TicTacHumanPlayer();

virtual bool MakeMove();

};

Реализация измененного заголовочного файла:

#include "TicTacHumanPlayer.h"

TicTacHumanPlayer::TicTacHumanPlayer()

{

}

TicTacHumanPlayer::~TicTacHumanPlayer()

{

}

bool TicTacHumanPlayer::MakeMove()

{

unsigned int col;

cout << "\n Игрок " << name << ",ваш ход...\n Введите столбец : ";

cin >> col;

if (this->board->CheckLegal(col))

{

this->board->SetCell(col, this->cellType);

return true;

}

return false;

}

Внесем изменения в класс менеджера нашей игры. Заменен заголовочный файл класса TicTacHumanPlayer на TicTacPlayer, и были исправлены переменные, относящиеся к предыдущему классу:

#pragma once

#include "TicTacBoard.h"

#include "TicTacPlayer.h"

class TicTacManager

{

private:

TicTacBoard\* board;

TicTacPlayer\* player1;

TicTacPlayer\* player2;

TicTacPlayer\* currentPlayer;

bool bGameFinished = false;

public:

TicTacManager();

virtual ~TicTacManager();

void ShowBoard();

void MakeMove();

bool isbGameFinished();

};

Добавим ИИ. Для этого создан на базе TicTacPlayer класс TicTacComputerPlayer:

#pragma once

#include "TicTacPlayer.h"

class TicTacComputerPlayer : public TicTacPlayer

{

public:

TicTacComputerPlayer();

virtual ~TicTacComputerPlayer();

virtual bool MakeMove();

};

Реализация класса:

bool TicTacComputerPlayer::MakeMove()

{

cout << "\n Ход компьютера:" << endl;

vector <TicTacBoardMonteCarloEvaluator\*> evaluators;

for (unsigned int i = 0; i < 7; i++)

if (this->board->CheckLegal(i))

evaluators.push\_back(new TicTacBoardMonteCarloEvaluator(this->board, 1000, (this->cellType == CellType\_X) ? CellType\_O : CellType\_X, i));

for (auto evaluator = evaluators.begin(); evaluator != evaluators.end(); evaluator++)

(\*evaluator)->Evaluate();

int biggestVictories = -1;

for (auto evaluator = evaluators.begin(); evaluator != evaluators.end(); evaluator++)

if (this->cellType == CellType\_X)

{

if ((\*evaluator)->GetVictories() > biggestVictories)

biggestVictories = (\*evaluator)->GetVictories();

}

else

{

if ((\*evaluator)->GetLosses() > biggestVictories)

biggestVictories = (\*evaluator)->GetLosses();

}

vector <TicTacBoardMonteCarloEvaluator\*> biggestWinEvaluators;

for (auto evaluator = evaluators.begin(); evaluator != evaluators.end(); evaluator++)

{

int numVictories;

if (this->cellType == CellType\_X)

numVictories = (\*evaluator)->GetVictories();

else

numVictories = (\*evaluator)->GetLosses();

if (numVictories == biggestVictories)

biggestWinEvaluators.push\_back((\*evaluator));

}

this->board->SetCell(biggestWinEvaluators[0]->GetColPos(), this->cellType);

for (auto evaluator = evaluators.begin(); evaluator != evaluators.end(); evaluator++)

delete(\*evaluator);

evaluators.clear();

biggestWinEvaluators.clear();

return true;

}

Добавили в менеджер игры возможность выбора типа игрока:

TicTacManager::TicTacManager()

{

string playerName;

int playerType;

this->board = new TicTacBoard(8, 7);

cout << " Введите тип игрока #1 (1 - Человек, любое другое число - Компьютер) : ";

cin >> playerType;

if (playerType == 1)

this->player1 = new TicTacHumanPlayer();

else

this->player1 = new TicTacComputerPlayer();

cout << " Введите тип игрока #2 (1 - Человек, любое другое число - Компьютер) : ";

cin >> playerType;

if (playerType == 1)

this->player2 = new TicTacHumanPlayer();

else

this->player2 = new TicTacComputerPlayer();

cin.ignore();

cout << " Введите имя игрока, играещего X : ";

getline(cin, playerName);

player1->SetupPlayer(playerName, CellType\_X);

cout << " Введите имя игрока, играещего O : ";

getline(cin, playerName);

player2->SetupPlayer(playerName, CellType\_O);

player1->SetBoard(this->board);

player2->SetBoard(this->board);

currentPlayer = player1;

}

Создаем класс CornersBoardMonteCarloEvaluator, объект которого принимает игровую доску и доигрывает ее случайными ходами, с помощью проделывания это множество раз, будут получены вероятности исходов.

Код заголовочного файла:

#pragma once

#include "TicTacBoard.h"

class TicTacBoardMonteCarloEvaluator

{

private:

TicTacBoard\* board;

int numGames;

int numVictories;

int numLosses;

int numDraws;

unsigned int col;

CellType startCellType;

public:

TicTacBoardMonteCarloEvaluator(TicTacBoard\* board, int numIterations, CellType startCellType, unsigned int col);

~TicTacBoardMonteCarloEvaluator();

void EvaluateBoard();

void Evaluate();

int GetVictories() { return numVictories; }

int GetLosses() { return numLosses; }

int GetDraws() { return numDraws; }

int GetColPos() { return col; }

};

#include "TicTacBoardMonteCarloEvaluator.h"

#include "TicTacRandomPlayer.h"

TicTacBoardMonteCarloEvaluator::TicTacBoardMonteCarloEvaluator(TicTacBoard\* board, int numIterations, CellType startCellType, unsigned int col)

{

this->board = board;

this->numGames = numIterations;

this->numDraws = 0;

this->numVictories = 0;

this->numLosses = 0;

this->col = col;

this->startCellType = startCellType;

}

TicTacBoardMonteCarloEvaluator::~TicTacBoardMonteCarloEvaluator()

{

}

void TicTacBoardMonteCarloEvaluator::EvaluateBoard()

{

TicTacBoard\* b = new TicTacBoard(this->board);

TicTacRandomPlayer\* player1 = new TicTacRandomPlayer();

TicTacRandomPlayer\* player2 = new TicTacRandomPlayer();

TicTacRandomPlayer\* currentPlayer;

bool bGameFinished = false;

player1->SetupPlayer("RandomX", CellType\_X);

player2->SetupPlayer("RandomO", CellType\_O);

player1->SetBoard(b);

player2->SetBoard(b);

currentPlayer = (this->startCellType == CellType\_X) ? player1 : player2;

if (b->CheckEndCondition())

{

if (b->isVictory())

if (currentPlayer == player1)

numLosses++;

else

numVictories++;

else

numDraws++;

bGameFinished = true;

return;

}

while (!bGameFinished)

{

while (!currentPlayer->MakeMove());

if (b->CheckEndCondition())

{

if (b->isVictory())

if (currentPlayer == player1)

numVictories++;

else

numLosses++;

else

numDraws++;

bGameFinished = true;

}

currentPlayer = (currentPlayer == player1) ? player2 : player1;

}

}

void TicTacBoardMonteCarloEvaluator::Evaluate()

{

for (int i = 0; i < numGames; i++)

EvaluateBoard();

}

Создан класс TicTacRandomPlayer, унаследованный от TicTacPlayer. Данный класс представляет собой сущность игрока, который делает случайные ходы.

Код заголовочного файла:

#pragma once

#include "TicTacPlayer.h"

class TicTacRandomPlayer : public TicTacPlayer

{

public:

TicTacRandomPlayer();

virtual ~TicTacRandomPlayer();

virtual bool MakeMove();

};

Реализация класса:

#include "TicTacRandomPlayer.h"

TicTacRandomPlayer::TicTacRandomPlayer()

{

}

TicTacRandomPlayer::~TicTacRandomPlayer()

{

}

bool TicTacRandomPlayer::MakeMove()

{

unsigned int col;

col = (unsigned int)(rand() % 7);

if (this->board->CheckLegal(col))

{

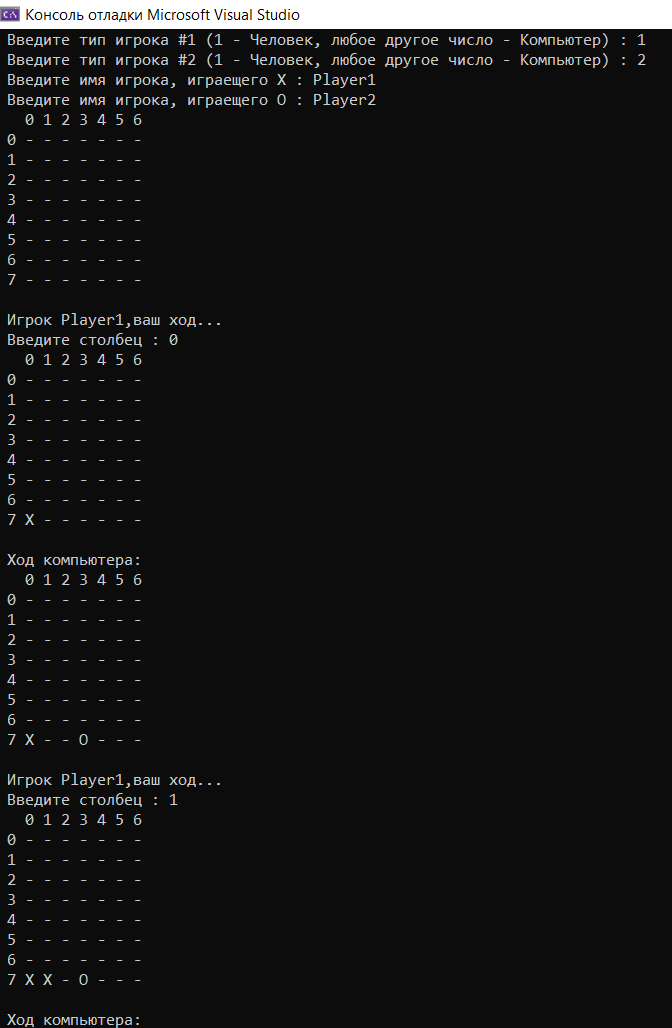
this->board->SetCell(col, this->cellType);

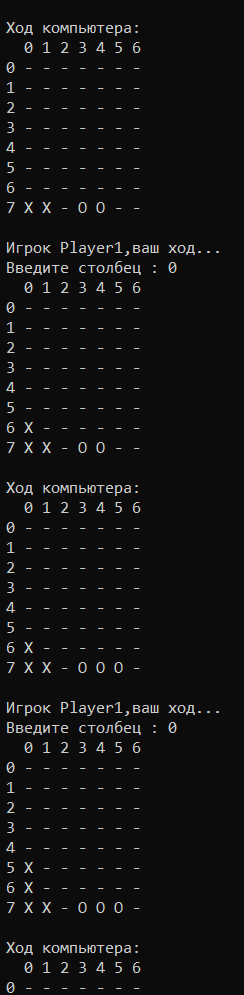
return true;

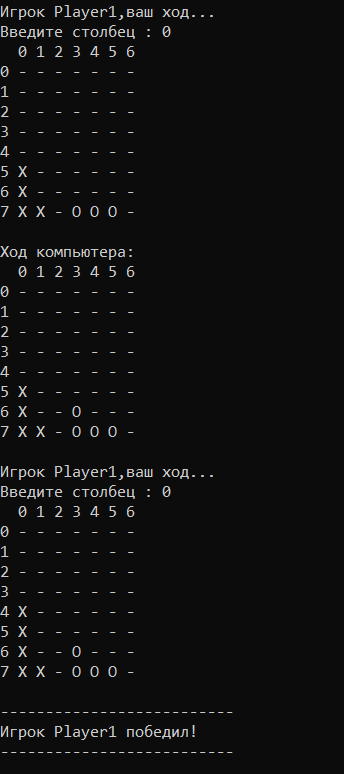
}

return false;

}







По итогам игры можно сделать вывод, что для игры в «4 в ряд 8х7» простая вероятностная модель игры подходит слабо, потому как требует серьезного усовершенствования,потому что оба компьютера особо не думают что можно заметить из последнего скрина, т.е. ставят О случайно. А из результатов игры человека и компьютера можно сделать другой вывод, игра сложная, т.к. очень многое зависит от ее правил, а также просчета ходов противника далеко вперед для выигрышного или, в крайнем случае, ничейного результата. Необходимо помнить, что возможностей выполнить разные ходы может быть очень много, что и влияет на результаты партий, и на быстродействие нашей системы. Обычно в таких случаях компьютеру «дают запомнить» все проигранные партии, и с помощью математического аппарата теории вероятностей добиваются расчёта оптимальных стратегий.

Вывод: реализован искусственный интеллект на базе вероятностного метода Монте-Карло.