МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра «Информационные технологии и общенаучные дисциплины»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**на лабораторную работу**

**по дисциплине «Алгоритмы и структуры данных»**

**Тема: Разработка компьютерной игры «крестики-нолики»**

Р.02069337. №23/36-9 ТЗ-00

Листов 19

|  |
| --- |
|  |
| Руководитель разработки: |
| кандидат технических наук, доцент |
| Шишкин Вадим Викторинович |
| «\_\_\_» \_\_\_\_\_\_\_\_ 202 г. |
|  |
| Исполнитель: |
| студентка гр. АИСТбд-21 |
| Крылова Наталья Владимировна |
| «\_\_\_» \_\_\_\_\_\_\_\_ 202 г. |

г. Ульяновск, 2024

**Содержание**

|  |  |
| --- | --- |
| Аннотация................................................................................................................. | 3 |
| Техническое задание................................................................................................ | 5 |
| Пояснительная записка............................................................................................ | 9 |
| Руководство программиста...................................................................................... | 13 |
| Текст программы...................................................................................................... | 15 |
|  |  |
|  |  |

**Аннотация**

В этом лабораторном отчете подробно описана реализация игры «Крестики-нолики» на языке Python с искусственным интеллектом, использующим Tkinter для графического пользовательского интерфейса. В программе используется простая эвристика для принятия решений ИИ. В отчете рассматривается реализация основной игровой логики, стратегии искусственного интеллекта и взаимодействия с пользователем.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра «Информационные технологии и общенаучные дисциплины»

**ТЕХНИЧЕСКОЕ ЗАДАНИЕ**

**на лабораторную работу**

**по дисциплине «Алгоритмы и структуры данных»**

**Тема: Разработка компьютерной игры «крестики-нолики»**

Р.02069337. №23/36-9 ТЗ-00

Листов 4

|  |
| --- |
|  |
| Исполнитель: |
| студентка гр. АИСТбд-21 |
| Крылова Наталья Владимировна |
| «\_\_\_» \_\_\_\_\_\_\_\_ 202 г. |

г. Ульяновск, 2024 г.

**Введение**

Настоящее техническое задание описывает разработку приложения “Крестики-нолики с ИИ”. Приложение реализует классическую игру “Крестики-нолики” с возможностью игры против искусственного интеллекта.

**Правила игры:** Два игрока по очереди ставят на игровом поле 3x3 знаки (крестики или нолики). Выигрывает игрок, первым выстроивший три своих знака в ряд (горизонтально, вертикально или по диагонали). Если все клетки заполнены, ни один игрок не победил, то объявляется ничья.

Приложение предоставляет следующие функциональные возможности:

* Игра против искусственного интеллекта, использующего алгоритм простой эвристики.
* Графический интерфейс пользователя (GUI) для интерактивной игры.
* Проверка корректности хода игрока.
* Отображение результата игры и хода игры.
* Возможность начать новую игру
* Возможность выбрать, за какую фигуру играть.

**1. Основания для разработки**

Учебный план направления 09.03.02 «Информационные системы и технологии»

**2. Требования к программе или программному изделию**

**2.1. Функциональное назначение**

Программа “Крестики-нолики с ИИ” предназначена для игры в классические “Крестики-нолики” против искусственного интеллекта. Программа автоматизирует процесс генерации ходов ИИ, проверки условий победы/поражения/ничьей, и отображения игрового процесса. Целевыми пользователями являются студенты, изучающие программирование ИИ, и все желающие поиграть в “Крестики-нолики” с сильным оппонентом.

**2.2. Требования к функциональным характеристикам**

2.2.1 Требования к структуре приложения

Программа имеет модульную структуру.

Модуль графического интерфейса: отвечает за визуальное отображение игрового поля и взаимодействие с пользователем.

Модуль игровой логики: реализует правила игры “Крестики-нолики”, определяет победителя и ничью.

Модуль ИИ: содержит алгоритм простой эвристики для генерации ходов ИИ.

2.2.2 Требования к составу функций приложения

Инициализация игры: Создание нового игрового поля. Выбор пользователя (X или O).

2.2.3 Требования к организации информационного обеспечения, входных и выходных данных

Пользовательский интерфейс: Должен быть интуитивным и удобным для использования. Игровое поле должно быть визуально понятно. Отображение текущего игрока, результатов игры и сообщений об ошибках.

Входные данные: Выбор пользователя (X или O), координаты хода игрока (строка, столбец).

Выходные данные: Визуальное отображение игрового поля, сообщения о ходе игры, результат игры.

Обработка хода человека: Прием координат хода от игрока, проверка корректности, обновление игрового поля.

Генерация хода ИИ: Использование алгоритма простой эвристики для выбора оптимального хода.

Проверка окончания игры: Определение победы, поражения или ничьей после каждого хода.

Отображение результатов: Информирование пользователя о результате игры (победа, поражение, ничья) на игровом поле или в отдельном окне.

Перезапуск игры: Опция для начала новой игры.

**2.3. Требования к надежности**

Программа должна быть устойчива к ошибкам ввода пользователя, обрабатывая некорректный ввод с соответствующими сообщениями. Программа не должна вызывать сбои в системе.

**2.4. Требования к информационной и программной совместимости**

* Версия операционной системы: Windows 10
* Используемые библиотеки: tkinter, random
* Язык Python: 3.11.9
* Используемая среда разработки: Visual Studio Code

**2.5. Требования к маркировке и упаковке**

Определяются заданием на лабораторную работу

**2.6. Требования к транспортированию и хранению**

2.6.1 Условия транспортирования

Требования к условиям транспортирования не предъявляются

2.6.2 Условия хранения

Требования к условиям транспортирования не предъявляются

2.6.3 Сроки хранения

Срок хранения – до окончания срока учебы

**3. Требования к программной документации**

Определяются заданием на лабораторную работу

**4. Стадии и этапы разработки**

Определяются заданием на лабораторную работу

**5. Порядок контроля и приемки**

Определяются заданием на лабораторную работу

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

федеральное государственное бюджетное образовательное учреждение высшего образования

**«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Кафедра «Информационные технологии и общенаучные дисциплины»

**Лабораторная работа**

**по дисциплине «Алгоритмы и структуры данных»**

**Тема: Разработка компьютерной игры «крестики-нолики»**

**Пояснительная записка**

Р.02069337. №23/36-9 ТЗ-00

Листов 3

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
| Исполнитель: |
| студентка гр. АИСТбд-21 |
| Крылова Наталья Владимировна |
| «\_\_\_» \_\_\_\_\_\_\_\_ 202 г. |

г. Ульяновск, 2024 г.

**Введение**

Данная пояснительная записка описывает разработку приложения “Крестики-нолики с ИИ”, реализующего классическую игру “Крестики-нолики” с использованием алгоритма простой эвристики для искусственного интеллекта. Приложение разработано на языке программирования Python с использованием библиотеки Tkinter для создания графического интерфейса. Цель работы – демонстрация реализации алгоритма простой эвристики для создания игры с интеллектуальным оппонентом.

**1. Проектная часть**

**1.1. Постановка задачи**

Задача заключается в разработке приложения “Крестики-нолики с ИИ” в соответствии с требованиями технического задания.

**1.2. Математическая модель**

В основе алгоритма ИИ лежит алгоритм простой эвристики.Он оценивает возможные ходы на основе следующих критериев: Блокировка: ИИ пытается заблокировать ходы игрока, создавая угрозы выигрыша (например, если игрок сделал два хода в одном ряду, ИИ попытается поставить свой ход в тот же ряд, чтобы заблокировать возможную победу игрока). Создание собственных угроз: ИИ также пытается создать свои собственные угрозы выигрыша, ставя ходы таким образом, чтобы создать два или более хода в одном ряду, столбце или диагонали. Центральная позиция: Если ни один из вышеперечисленных критериев не применим, ИИ пытается занять центральную позицию на доске (если она свободна), так как это дает ему наилучшие шансы на выигрыш. Углы: Если центральная позиция занята, ИИ пытается занять один из углов доски. Стороны: Если все углы заняты, ИИ пытается занять одну из сторон доски. ИИ выбирает ход, который наилучшим образом соответствует этим критериям, в следующем порядке: блокировка, создание собственных угроз, центральная позиция, углы, стороны. Функция bot\_move в представленном коде реализует данный алгоритм.

**1.3. Архитектура и алгоритмы**

1.3.1 Архитектура

Приложение состоит из трех основных модулей:

Модуль графического интерфейса (GUI): Создан с помощью Tkinter, отвечает за визуальное представление игрового поля и обработку пользовательского ввода. Функция new\_game инициализирует GUI.

Модуль игровой логики: Реализует правила игры “Крестики-нолики”, включая проверку условий победы, поражения и ничьей. Функция winner определяет, чем закончилась игра: победой бота, победой игрока или ничьей.

Модуль ИИ: Включает в себя реализацию алгоритма простой эвристики. Функция bot\_move выполняет ход ИИ.

1.3.2 Алгоритм простой эвристики

Алгоритм простой эвристики оценивает возможные ходы на основе следующих критериев:

Блокировка: ИИ пытается заблокировать ходы игрока, создавая угрозы выигрыша (например, если игрок сделал два хода в одном ряду, ИИ попытается поставить свой ход в тот же ряд, чтобы заблокировать возможную победу игрока).

Создание собственных угроз: ИИ также пытается создать свои собственные угрозы выигрыша, ставя ходы таким образом, чтобы создать два или более хода в одном ряду, столбце или диагонали.

Центральная позиция: Если ни один из вышеперечисленных критериев не применим, ИИ пытается занять центральную позицию на доске (если она свободна), так как это дает ему наилучшие шансы на выигрыш.

Углы: Если центральная позиция занята, ИИ пытается занять один из углов доски.

Стороны: Если все углы заняты, ИИ пытается занять одну из сторон доски.

ИИ выбирает ход, который наилучшим образом соответствует этим критериям, в следующем порядке: блокировка, создание собственных угроз, центральная позиция, углы, стороны.

**1.4. Тестирование**

1.4.1 Описание интеллектуальной карты приложения

Тестирования были проведены: победа ИИ, ничья, обработка некорректного ввода.

1.4.2 Цель тестирования

Проверка корректности работы всех модулей приложения, включая алгоритм Minimax, и отсутствие критических ошибок.

1.4.3 Методика тестирования

Использовалось ручное тестирование различных сценариев игры, включая победы ИИ и ничью.

1.4.4 Проведенные тесты

Были проведены полноценные игры с сильной позицией игрока (проверка, что бот не может проиграть в такой ситуации), а также с сильной позицией бота (в этом случае бот выигрывал)

1.4.5 Выводы

Тестирование показало корректную работу приложения и реализованного алгоритма простой эвристики.

**2. Источники**

1. Шишкин В.В., Афонин Д.С. Разработка логических компьютерных игр с графическим интерфейсом в среде Питон. - УлГТУ, 2023
2. Воронина В.В., Шишкин В.В. Компьютерная графика.-УлГТУ, 2023

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

федеральное государственное бюджетное образовательное учреждение высшего образования

**«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Кафедра «Информационные технологии и общенаучные дисциплины»

**Лабораторная работа**

**по дисциплине «Алгоритмы и структуры данных»**

**Тема: Разработка компьютерной игры «крестики-нолики»**

**Руководство программиста**

Р.02069337. №23/36-9 ТЗ-00

Листов 3

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
| Исполнитель: |
| студентка гр. АИСТбд-21 |
| Крылова Наталья Владимировна |
| «\_\_\_» \_\_\_\_\_\_\_\_ 202 г. |

г. Ульяновск, 2024 г.

**1. Назначение и условия применения программы**

**1.1 Назначение и функции, выполняемые приложением**

Приложение “Крестики-нолики с ИИ” предназначено для игры в классическую игру “Крестики-нолики” против искусственного интеллекта (ИИ). Приложение реализует алгоритм простой эвристики для принятия решений ИИ, обеспечивает интерактивное взаимодействие с пользователем через графический интерфейс и предоставляет результаты игры (победа, поражение, ничья).

**1.2 Условия, необходимые для использования приложения**

Для работы приложения необходимы:

Операционная система: Windows 10 и выше.

Язык программирования: Python 3.11 или выше.

Библиотеки: Tkinter.

**2. Характеристики программы**

**2.1 Характеристики приложения**

Приложение содержит 138 строк кода и использует следующие структуры данных: массив для представления игрового поля (games), переменные для отслеживания текущего игрока (choice), состояния игры (win), и оптимального хода (index).

**2.2 Особенности реализации приложения**

Алгоритм простой эвристики реализован с помощью функции bot\_move, которая выполняет обход всех возможных вариантов расстановки фигур на игровом поле.Графический интерфейс создан с помощью библиотеки Tkinter. Функция winner определяет исход игры путем поиска текущей комбинации среди всех возможных выигрышных комбинаций. Обработка пользовательского ввода осуществляется через обработчики событий для кнопок игрового поля (click).

**3. Обращение к программе**

Приложение запускается путем выполнения основного скрипта на Python. Взаимодействие с программой осуществляется через графический интерфейс, представляющий собой игровое поле с кнопками. Пользователь выбирает, играть за “X” или “O”, после чего начинается игра. ИИ использует алгоритм простой эвристики для выбора своего хода.

**4. Сообщения**

Cообщения, выдаваемые по результатам контроля корректности ввода/вывода:

* Ничья! (если игра окончилась ничьей)
* Победа! Указывается кто победил игрок или бот (если игра окончилась победой бота или игрока)
* Хотите сыграть ещё раз? (конец игры, возможность начать новую игру)

**Текст программы**

|  |
| --- |
| from tkinter import \*  import random  from tkinter.messagebox import \*  root = Tk()  root.title('Крестики Нолики')  root.geometry('300x295+620+180')  games=[]  choice = 0  combinations = [(0, 1, 2), (3, 4, 5), (6, 7, 8), (0, 3, 6), (1, 4, 7), (2, 5, 8), (0, 4, 8), (2, 4, 6)]  def new\_game():  global run\_game, win, condition, games, button1  if games!=[]:  for i in range(3):  for j in range(3): games[i][j].grid\_forget()  button1.grid\_forget()  games=[]  return fig()  run\_game = True  condition = [None] \* 9  win = None  games = []  root.columnconfigure(index=2,weight=50)  for i in range(3):  row = []  for j in range(3):  button = Button(root, text="", width=10, height=5, command=lambda row=i, col=j: click(row, col))  button.grid(row=i, column=j,sticky='ew')  row.append(button)  games.append(row)  button1 = Button(root,font='9', text="Новая игра", command=new\_game)  button1.grid(row = 3, column=0,columnspan=3, sticky='nswe')  if choice == 0:bot\_move()  def add\_x(row, col):  games[row][col]['text'] = 'X'  games[row][col]['state'] = 'disabled'  def add\_o(row, col):  games[row][col]['text'] = 'O'  games[row][col]['state'] = 'disabled'  def click(row, col):  global choice  if run\_game:  index = row \* 3 + col  if condition[index] is None:  if choice == 1:  condition[index] = 1  add\_x(row, col)  if winner():  end\_game()  else:  bot\_move()  if winner():  end\_game()  elif choice == 0:  condition[index] = 0  print(condition)  add\_o(row, col)  if winner():end\_game()  else:  bot\_move()  if winner():end\_game()  else:new\_game()  def bot\_move():  index = None  for i in combinations:  variants = (([condition[i[0]], condition[i[1]], condition[i[2]]]))  if variants.count(1-choice) == 2 and variants.count(None) == 1:  index = i[variants.index(None)]  break  if index is None:  for i in combinations:  variants = (([condition[i[0]], condition[i[1]], condition[i[2]]]))  if variants.count(choice) == 2 and variants.count(None) == 1:  index = i[variants.index(None)]  break  if index is None:  for i in combinations:  variants = (([condition[i[0]], condition[i[1]], condition[i[2]]]))  if variants.count(1-choice) == 1 and variants.count(None) == 2:  index = i[variants.index(None)]  break  if index is None:  if condition[4] is None:  index = 4  if index is None:  empty\_indexes = []  for i in range(0, 9, 2):  if condition[i] is None:  empty\_indexes.append(i)  if empty\_indexes:  index = random.choice(empty\_indexes)  if index is None:  empty\_indexes = []  for index, el in enumerate(condition):  if el is None:  empty\_indexes.append(index)  if empty\_indexes:  index = random.choice(empty\_indexes)  condition[index] = 1-choice  row = index // 3  col = index % 3  if choice==0: add\_x(row, col)  else: add\_o(row, col)  def winner():  global win  variants = []  for i in combinations:variants.append([condition[i[0]], condition[i[1]], condition[i[2]]])  if [choice] \* 3 in variants:win = 'Ты ПОБЕДИЛ!'  elif [1-choice] \* 3 in variants:win = 'Бот Выиграл'  elif None not in condition:win = 'Ничья'  return win  def end\_game():  global run\_game,games  run\_game = False  for row in games:  for button in row:button['state'] = 'disabled'  showinfo("Игра окончена", win)  for i in range(3):  for j in range(3):games[i][j].grid\_forget()  button1.grid\_forget()  games=[]  fig()  def choose\_x():  global choice  choice = 1  new\_game()  def choose\_o():  global choice  choice = 0  new\_game()  def fig():  root.columnconfigure(index=0,weight=50)  root.columnconfigure(index=1,weight=50)  root.columnconfigure(index=2,weight=0)  Label(root, text="Выберите фигуру:",font='9').grid(row=0, column=0, columnspan=2)  Button(root, text="X", font='9', command=choose\_x).grid(row = 1, column = 0, sticky='ew',)  Button(root, text="O", font='9', command=choose\_o).grid(row = 1, column = 1, sticky='ew')  fig()  root.mainloop() |