МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра «Измерительно-вычислительные комплексы»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | |  | |  |  |
| |  |  | | --- | --- | | ***Подп. и дата*** |  | | ***Инв. № дубл.*** |  | | ***Взам. инв. №*** |  | | ***Подп. и дата*** |  | | ***Инв. № подл*** |  | |  | | ТЕХНИЧЕСКОЕ ЗАДАНИЕ  на лабораторную работу №8  по дисциплине «Алгоритмы и структуры данных»  Тема «Управление договорами на аренду автомобилей» | | | | | | |
|  | | |  | |  | | | |
|  | | Исполнитель  студент гр. ИСТбд-21  Пирогов П.Е.  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2024 г. | | | |
|  | | 2024 | | | | | | |

**Введение**

Этот проект представляет собой графическое приложение для игры в "Крестики-нолики", реализованное на языке Python с использованием библиотеки Tkinter. Игра предоставляет два режима: игра между двумя игроками или игра против компьютера, где ИИ использует алгоритм Minimax для оптимального выбора ходов.

**Среда разработки**

Для работы с проектом необходимо установить следующие компоненты:

* **Python 3.x** — основной язык программирования.
* **Tkinter** — библиотека для создания графического интерфейса, которая включена в стандартную поставку Python.

Для корректной работы программы необходимо использовать актуальную версию Python и Tkinter.

**Структура проекта**

Проект состоит из одного файла, который включает в себя:

* Логику игры.
* Реализацию интерфейса.
* Алгоритм для работы с искусственным интеллектом.

**Описание основных классов и модулей**

**Класс TicTacToe**

Этот класс управляет всей логикой игры, включая обработку ходов игроков, взаимодействие с интерфейсом и алгоритм ИИ для игры против компьютера.

Основные методы класса:

* \_\_init\_\_(self): Инициализирует начальные параметры игры, создает интерфейс и игровое поле.
* initialize\_ui(self): Создает элементы пользовательского интерфейса, такие как кнопки для выбора режима игры.
* initialize\_board(self): Инициализирует игровое поле в виде сетки 3x3 с кнопками.
* create\_mode\_buttons(self): Создает кнопки для выбора режима игры (игрок против игрока или игрок против компьютера).
* on\_click(self, row, col): Обрабатывает клики по ячейкам игрового поля.
* make\_move(self, row, col, player): Обрабатывает ход игрока, обновляет поле и проверяет условия победы или ничьей.
* ai\_move(self): Выполняет ход компьютера, используя алгоритм Minimax для вычисления лучшего хода.
* minimax(self, depth, is\_maximizing): Реализует алгоритм Minimax, который помогает ИИ выбрать оптимальный ход.
* check\_win(self, player): Проверяет, есть ли победитель на поле.
* check\_draw(self): Проверяет, закончилась ли игра вничью.
* reset\_game(self): Сбрасывает игровое поле, устанавливает начальные параметры и готовит игру к новому запуску.
* start\_game(self): Запускает главный цикл игры.

**Детали реализации**

**1. Пользовательский интерфейс**

* Игровое поле представлено сеткой из 3x3 кнопок, каждая из которых представляет клетку игрового поля.
* Интерфейс включает кнопки для выбора режима игры ("Игрок против Игрока" или "Игрок против Компьютера").
* Есть кнопка для перезапуска игры после завершения текущей партии.

**2. Логика игры**

* Игра идет по очереди: игроки ставят свои символы ("X" или "O") в пустые клетки.
* Победитель определяется по правилам: три одинаковых символа в ряду, столбце или диагонали.
* Если все клетки заполнены, а победителя нет, объявляется ничья.

**3. Искусственный интеллект**

* Алгоритм Minimax используется для выбора оптимального хода компьютера.
* Алгоритм просчитывает все возможные ходы и оценивает их, выбирая тот, который приносит наибольшую вероятность победы.

**Расширение функциональности**

**1. Добавление новых режимов игры**

* Новые режимы можно реализовать, добавив дополнительные методы для новых типов игр и соответствующие кнопки в интерфейсе.

**2. Оптимизация ИИ**

* В алгоритм Minimax можно добавить альфа-бета отсечение для улучшения производительности, что ускорит процесс принятия решения.

**3. Добавление статистики**

* Можно внедрить систему подсчёта побед и ничьих для каждого игрока, а также добавить возможность хранения статистики игр.

**Тестирование и отладка**

**1. Основные тесты:**

* Проверка корректности определения победы для всех возможных комбинаций.
* Тестирование логики игры в различных режимах: два игрока против друг друга и игрок против компьютера.
* Проверка работы кнопки "Рестарт" для перезапуска игры.

**2. Отладка:**

* Для отладки рекомендуется использовать встроенные инструменты отладки в IDE (например, PyCharm).
* Логирование ходов и состояний игры также может помочь в поиске ошибок.

**Заключение**

Данный проект демонстрирует реализацию классической игры "Крестики-нолики" с возможностью игры против компьютера с использованием алгоритма Minimax. Он предоставляет основу для дальнейших улучшений, таких как добавление новых игровых режимов, оптимизация искусственного интеллекта и расширение функционала.