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*** |  | |  | | **Пояснительная записка**  на лабораторную работу №9  по дисциплине «Алгоритмы и структуры данных»  **Тема** «Разработка компьютерной игры Крестики Нолики» | | | | | | |
|  | | |  | |  | | | |
|  | | Исполнитель  студент гр. ИСТбд-21  Салова А.С.  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2024 г. | | | |
|  | | 2024 | | | | | | |

**Введение**

Разработанное приложение представляет собой игру "Крестики-нолики", реализованную на языке Python с использованием библиотеки **Tkinter** для создания графического интерфейса.  
Цель программы — предоставить пользователям возможность играть как друг с другом, так и против компьютера, демонстрируя простую и интуитивно понятную игровую логику.

**Основания для разработки**

Программа создана в рамках лабораторной работы по учебному плану направления "Информационные системы и технологии".

**Постановка задачи**

**Цель разработки:**  
Создание игры "Крестики-нолики" с двумя режимами игры (против компьютера или другого игрока), реализацией алгоритма минимакс и удобным графическим интерфейсом.

**Структура программы**

**Основные функции:**

1. **initialize\_ui** — создание и настройка пользовательского интерфейса.
2. **initialize\_board** — подготовка игрового поля (очистка и инициализация).
3. **set\_first\_move** — выбор первого игрока.
4. **player\_vs\_player** — режим игры "игрок против игрока".
5. **player\_vs\_ai** — режим игры "игрок против компьютера".
6. **on\_click** — обработка нажатий на кнопки игрового поля.
7. **make\_move** — выполнение хода текущего игрока.
8. **ai\_move** — ход компьютера с использованием алгоритма минимакс.
9. **minimax** — реализация алгоритма минимакс для оптимизации игры ИИ.
10. **check\_win** — проверка наличия победителя.
11. **check\_draw** — проверка, наступила ли ничья.
12. **reset\_game** — перезапуск игры.

**Функциональное назначение программы**

1. Создание игрового поля размером 3x3.
2. Реализация двух режимов игры: PvP и PvE.
3. Отображение текущего статуса игры и хода.
4. Возможность перезапуска игры.
5. Выбор первого игрока перед началом игры.

**Проектная часть**

**1.1. Математические методы**

Для реализации компьютерного игрока используется алгоритм минимакс.  
Алгоритм позволяет просчитывать возможные ходы и их последствия, выбирая оптимальный ход для компьютера.

**1.2. Архитектура программы**

**1.2.1. Основные элементы:**

Программа построена на основе класса **TicTacToe**, который содержит:

* **Игровое поле**: двумерный массив (self.board).
* **Интерфейсные кнопки**: кнопки управления и игрового поля (self.buttons).
* **Режим игры**: переменная, определяющая режим игры (PvP или PvE).
* **Текущий игрок**: идентификатор текущего игрока (self.current\_player).
* **Состояние игры**: флаг завершения игры (self.game\_over).

**1.2.2. Алгоритмы:**

* **Минимакс**: использует рекурсивный анализ всех возможных комбинаций ходов.
* **Проверка победы**: определяет наличие победителя, проверяя линии, столбцы и диагонали.
* **Проверка ничьей**: определяет, заполнено ли поле, если победитель не найден.

**Стадии и этапы разработки**

1. **Определение требований**:  
   Постановка задач и выделение ключевых функций программы.
2. **Проектирование**:  
   Создание структуры программы и пользовательского интерфейса.
3. **Реализация**:  
   Написание кода программы, включая графический интерфейс и игровую логику.
4. **Тестирование**:  
   Проверка функциональности программы на различных сценариях.
5. **Оптимизация**:  
   Улучшение производительности и отзывчивости интерфейса.
6. **Документирование**:  
   Создание пояснительной записки и добавление комментариев в код.

**Тестирование**

**Цель тестирования**

Проверка правильности работы всех компонентов программы:

* Игровой логики.
* Графического интерфейса.
* Искусственного интеллекта.

**Методика тестирования**

Тестирование проводилось методом "черного ящика" для проверки функционала игры при различных сценариях.

**Проведённые тесты:**

1. Модульное тестирование ключевых функций:
   * Проверка алгоритма минимакс.
   * Проверка работы механизма смены ходов.
2. Интеграционное тестирование:
   * Проверка взаимодействия игровых компонентов.
3. Функциональное тестирование:
   * Проверка работы режимов игры.
   * Тестирование на корректность определения победителя.
4. Проверка пользовательского интерфейса:
   * Удобство использования и отзывчивость кнопок.
   * Проверка функциональности кнопок перезапуска и переключения режимов.

**Результаты тестирования**

* Исправлены ошибки в алгоритме минимакс, связанные с некорректным выбором хода.
* Оптимизирована проверка победы.
* Улучшен пользовательский интерфейс для повышения удобства.

**Заключение**

Разработанное приложение "Крестики-нолики" успешно реализует поставленные задачи:

* Поддерживаются два режима игры: против игрока и против компьютера.
* Реализован удобный графический интерфейс.
* Искусственный интеллект, построенный на основе алгоритма минимакс, обеспечивает высокий уровень сложности игры.

Программа может быть использована в учебных целях для демонстрации принципов работы с **tkinter**, реализации алгоритма минимакс и построения логики игр.

**Источники, использованные при разработке**

1. Лутц М. *Изучаем Python*. – СПб.: Символ-Плюс, 2011.
2. **Документация Tkinter** [Электронный ресурс]. Официальная документация Python:  
   <https://docs.python.org/3/library/tkinter.html>.
3. Рассел С., Норвиг П. *Искусственный интеллект: современный подход*. – М.: Вильямс, 2006.
4. Саммерфилд М. *Python на практике*. – М.: ДМК Пресс, 2014.