МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра «Измерительно-вычислительные комплексы»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

на курсовую работу

по дисциплине «Алгоритмы и структуры данных»

Тема Компьютерная логическая игра Сиджа

**Инв. № подл.**

**Подп. и дата**

**Взам. инв. №**

**Инв. № дубл.**

**Подп. и дата**

Р.0206933722/31217 ТЗ-01

Листов 32

Руководитель разработки:

кандидат технических наук, доцент кафедры «Измерительно-вычислительные комплексы», *Шишкин Вадим Викторинович*

« » 2023 г.

Исполнитель:

студент гр. ИСТбд-21

*Саранцев Евгений Андреевич*

« » 2023 г.

2023

Содержание

Аннотация……...…………………………………………………………3

Техническое задание…………………………………………………...4

Пояснительная записка...……………………………………………...9

Руководство пользователя……………………………….…………...14

Текст программы…..…………………………………………………...23

**Аннотация**

Задание на курсовую работу: разработка компьютерной логической игры «Сиджа».

Суть задания: необходимо разработать приложение, которое соответствует правилам настольной игры «Сиджа». Оно должно представлять собой окно, представляющую шахматную доску с интуитивно понятные интерфейсом и основным функционалом.

Ключевые функции приложения:

1. Авторизация/регистрация пользователя.
2. Отображение игрового поля.
3. Проверка ходов игрока.
4. Ходы искусственного интеллекта.
5. Объявление результатов игры.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра «Измерительно-вычислительные комплексы»

ТЕХНИЧЕСКОЕ ЗАДАНИЕ

на курсовую работу

по дисциплине «Алгоритмы и структуры данных»

Тема Компьютерная логическая игра Сиджа

**Инв. № подл.**

**Подп. и дата**

**Взам. инв. №**

**Инв. № дубл.**

**Подп. и дата**

Р.0206933722/31217 ТЗ-01

Листов 5

Исполнитель:

студент гр. ИСТбд-21

*Саранцев Евгений Андреевич*

« » 2023 г.

2023

**Введение**

Приложение должно соответствовать следующим правилам игры Сиджа.

Игра ведется между двумя соперниками (пользователь-компьютер) фишками разного цвета на квадратном поле размером 5х5 клеток. В начальной стадии у каждого игрока 12 фишек и пустое поле.  
Каждый игрок делает ход и размещает по одной фишке за раз в любые пустые ячейки по своему выбору, кроме центральной ячейки. Фишки расставляются на поле по 1 штуки за ход, пока не будет заполнено всё поле, кроме центральной клетки.

После того, как все фишки расставлены, начинается вторая стадия игры. Игроки по очереди перемещают по одной, из занятой клетки в пустую. Каждый игрок пытался зажать фишку соперника между двумя своими фишками. Если фишка противника оказывается зажатой между двумя фишками игроков, то эта фигура удаляется с доски. Фишки могут двигаться по горизонтали или вертикали.  
Захваченная фишка выбивается только в том случае, если игрок сделал ход, чтобы зажать ее одной из своих фигур. Если фишка игрока САМА входит в клетку, которая размещается между двумя фишками противника, она не снимается.

Фишки могут перемещаться только на одну клетку за раз и не могут пропускать клетки или перепрыгивать через другие фишки, будь то свои или противника.

Цель игры – удалить все фишки противника с поля зажав их между своими.

Игра идет до того момента пока у противника не останется ни одной фишки.

Объекты игры:

Игровое поле. Описание: Квадратное поле стандартным размером 5х5 клеток, на котором размещаются фишки игроков.

Фишка игрока. Описание: Фишка белого или черного цвета, размещаемая на игровом поле.

**1. Основания для разработки**

В качестве оснований для разработки указывается учебный план направления 09.03.02 «Информационные системы и технологии» и распоряжение по факультету.

**2. Требования к программе**

**2.1. Функциональное назначение**

* Создание компьютерной игры "Сиджа" с возможностью игры против компьютера.
* Реализация игрового интерфейса, позволяющего игрокам делать ходы и отображать состояние игрового поля.
* Разработка алгоритма компьютера для противостояния игроку.
* Оценка победы или поражения игроков и определение конца игры.
* Обеспечение удобного пользовательского опыта и интерфейса для взаимодействия с игрой.
* Отображение игрового поля с фишками.
* Обработка ходов игроков и обновление игрового состояния.
* Оценка текущей ситуации на игровом поле.
* Проверка окончания игры и объявление победителя.

Группы пользователей:

* Игрок: Человек, играющий против компьютера.
* Компьютер: Алгоритм, который играет против человека.

**2.2 Требования к функциональным характеристикам**

2.2.1 Требования к структуре приложения

* Игровой движок: Модуль, отвечающий за логику игры, обработку ходов, проверку правил игры и вычисление победителя.
* Интерфейс пользователя: Модуль, обеспечивающий взаимодействие с игроком, отображение игрового поля, обработку пользовательских действий (нажатия кнопок, ввод данных и т.д.).
* Компьютер: Модуль, реализующий алгоритм компьютера, который принимает решения о ходах на основе текущего состояния игры и стратегии.
* Графический интерфейс пользователя: Обеспечивает отображение игрового поля с фишками, кнопки для совершения ходов, отображение текущего состояния игры

2.2.2 Требования к составу функций приложения

Для игрока:

Отображение игрового поля:

* Отображение текущего состояния игрового поля с расположением фишек.

Совершение хода:

* Возможность выбора фишки и клетки для совершения хода.
* Обновление состояния игрового поля после выполнения хода.

Проверка окончания игры:

* Проверка условий окончания игры.
* Определение победителя и отображение соответствующего сообщения.

Для компьютера:

Алгоритм принятия решений:

* Разработка алгоритма, основанного на стратегии игры, для выбора лучшего хода компьютерного игрока.
* Оценка текущего состояния игры и возможных ходов.

2.2.3 Требования к организации информационного обеспечения, входных и выходных данных

Требования к пользовательскому интерфейсу:

* Графический интерфейс должен быть привлекательным и простым для понимания пользователем.
* Использование понятных и наглядных элементов управления.
* Четкое отображение игрового поля с фишками.
* Мгновенная реакция на действия пользователя без заметных задержек.
* Предоставление информации об ошибках или некорректных действиях пользователя.

**2.3 Требования к надёжности**

* Приложение должно быть устойчивым к возможным сбоям, ошибкам и исключительным ситуациям.
* Обработка исключительных ситуаций и ошибок с помощью механизмов обработки исключений.

**2.4 Требования к информационной и программной совместимости**

* ОС Windows
* Версия Python: 3.11
* Среда разработки: PyCharm Community Edition 2022.3.2

**2.5 Требования к маркировке и упаковке**

Определяются заданием на курсовую 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 хранения не предъявляются.

2.6 3 Сроки хранения

Срок хранения – до июля 2024 года.

**3. Требования к программной документации**

Определяются заданием на курсовую работу.

**4. Стадии и этапы разработки**

Определяются заданием на курсовую работу.

**5. Порядок контроля и приёмки**

Определяются заданием на курсовую работу.

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

федеральное государственное бюджетное образовательное учреждение

высшего образования

**«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Кафедра «Измерительно-вычислительные комплексы»

**Курсовая работа**

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

**Тема** Компьютерная логическая игра Сиджа

**Пояснительная записка**

Р.0206933722/31217 ПЗ-01

Листов 5

**Исполнитель**:

студент гр. ИСТбд-21

*Саранцев Евгений Андреевич*

« » 2023 г.

2023

**Введение**

Название разрабатываемого приложения — "Сиджа". Это приложение представляет собой электронную версию игры "Сиджа", доступную на персональных компьютерах.

Приложение обладает удобным и интуитивно понятным интерфейсом, позволяющим игрокам легко размещать свои фишки на игровом поле и применять стратегические решения.

**1. Проектная часть**

**1.1 Постановка задачи на разработку приложения**

Определяется заданием на курсовую работу. Детализируется в разработанном техническом задании (приложение 1).

**1.2 Математические методы**

При разработке "Сиджа" был выбран подход, основанный на математической модели игры "Сиджа". Математическая модель позволяет точно определить логику ходов, правила игры и алгоритмы принятия решений компьютерного противника. Такой подход обеспечивает точность и надежность реализации игры, а также позволяет создать интеллектуального компьютерного противника.

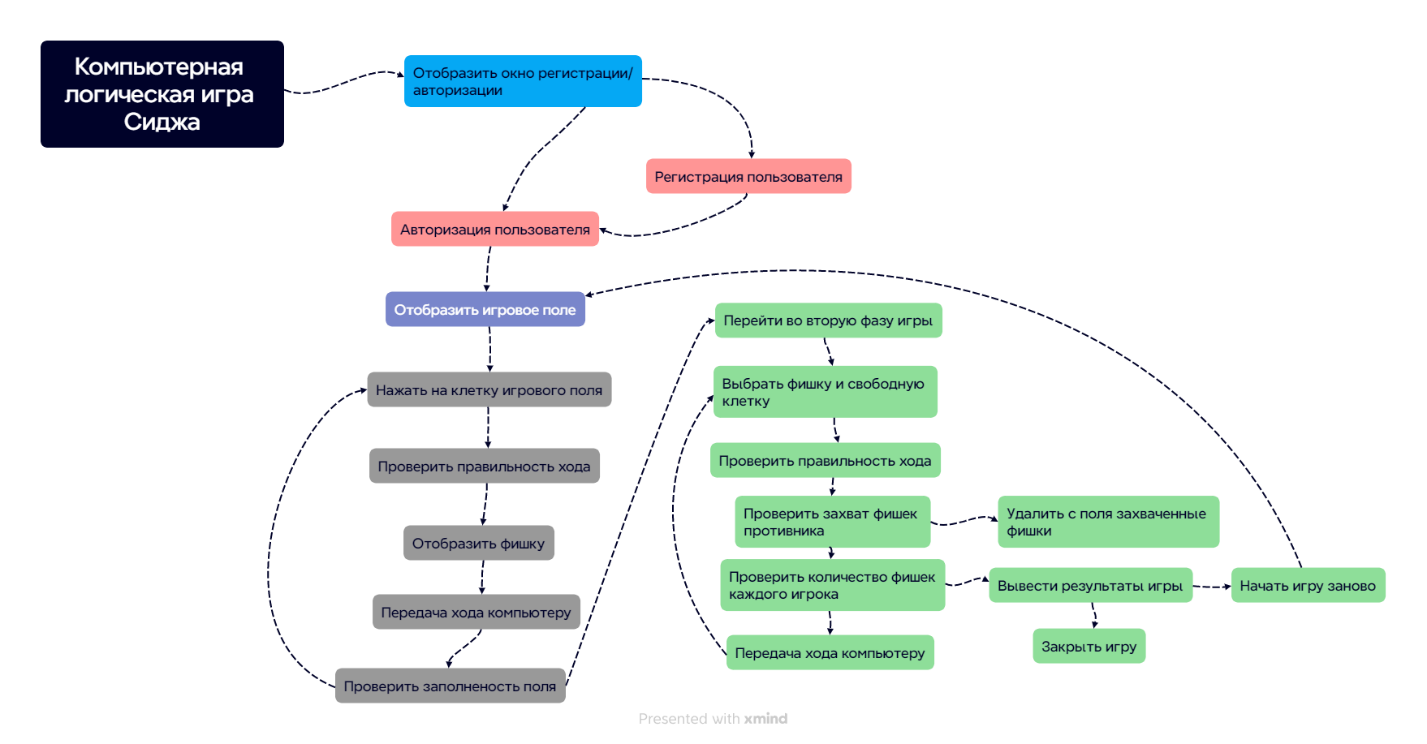
**1.3 Архитектура и алгоритмы**

1.3.1. Архитектура

Классы и структура данных:

Game: этот класс представляет игру «Сиджа». Он содержит методы для управления игровым процессом, хранит состояние игрового поля.

Authentication: этот класс представляет окно авторизации/регистрации пользователя. Он содержит методы для отображения окна, проверки данных о зарегистрированных пользователях и последующей авторизации, а также метод регистрации пользователя.

**1.4 Тестирование**

1.4.1 Описание отчета о тестировании

Отчет о тестировании игры "Сиджа" предоставляет обзор текущего состояния игры и качества ее реализации. Он включает информацию о проведенных функциональных тестах, тестировании пользовательского интерфейса, проверке граничных значений, совместимости, производительности и безопасности. Каждый из этих аспектов тестирования подробно описывается в отчете.

Основная цель отчета о тестировании игры "Сиджа" - предоставить полную картину о ее качестве и выявленных проблемах. Отчет содержит рекомендации по исправлению проблем и улучшению функциональности игры.

1.4.2 Цель тестирования

Проверить функциональность, корректность ввода/вывода и общую стабильность приложения "Сиджа".

1.4.3 Методика тестирования

Модульное тестирование: в этом методе можно проводится тестирование отдельных модулей игры, таких как логика ходов, правила игры, алгоритмы принятия решений компьютерного противника и другие важные компоненты. Это позволит убедиться в правильной работе каждого модуля и их соответствии требованиям.

Интеграционное тестирование: здесь проверяется взаимодействие различных модулей игры между собой. Например, как правильно обновляется игровое поле при ходе игрока или компьютера, как корректно обновляется счет игры и т.д. Целью является выявление возможных ошибок во взаимодействии компонентов игры.

Функциональное тестирование: этот метод позволяет проверить функциональность игры в соответствии с требованиями. Проверить правильность подсчета очков, корректность применения правил игры и другие функциональные аспекты.

Графическое тестирование: этот метод предполагает проверку корректности отображения графического интерфейса игры. Это включает проверку расположения элементов на экране, соответствие цветов и шрифтов, и других визуальных аспектов игры.

1.4.4 Проведенные тесты

Тестирование функциональности:

- Цель: проверить основную функциональность игры "Сиджа".

- Шаги выполнения:

1. Открыть приложение "Сиджа".

2. Зарегистрировать нового пользователя.

3. Начать игру.

4. Выполнить несколько ходов.

5. Завершить игру.

Ожидаемые результаты:

- Пользователь успешно зарегистрирован.

- Игра успешно запускается.

- Игровое поле отображается корректно.

- Игрок может совершать ходы в соответствии с правилами игры.

- Результаты игры отображаются правильно.

- Фактические результаты: Все ожидаемые результаты подтверждены.

Тестирование корректности ввода/вывода:

Цель: проверить корректность обработки пользовательского ввода и вывода информации.

- Шаги выполнения:

1. Оставить окно регистрации пустым.

2. Выбрать клетку во время хода, не соответствующую правилам игры.

Ожидаемые результаты:

- Если оставить поля в окне регистрации пустыми, то выводится сообщение об ошибке.

- При выборе клетки во время хода, не соответствующей правилам игры, ход не производится.

Фактические результаты: все ожидаемые результаты подтверждены.

1.3.5 Выводы

Приложение "Сиджа" демонстрирует стабильную работу и правильную функциональность в большинстве случаев.

Необходимо добавить подсказку, когда игрок пытается сделать ход, не соответствующий правилам игры.

**2. Источники, использованные при разработке**

1. Сиджа [Электронный ресурс] – Режим доступа: <https://lotos-khv.ru/game/games/sidja.pdf>
2. Документация библиотеки tkinter [Электронный ресурс] – Режим доступа: <https://tkdocs.com/index.html>
3. Большой учебник по тестированию [Электронный ресурс]. – Режим доступа: <https://qarocks.ru/big-software-testing-textbook/#fundamentals>

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

федеральное государственное бюджетное образовательное учреждение

высшего образования

**«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Кафедра «Измерительно-вычислительные комплексы»

**Курсовая работа**

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

**Тема** Компьютерная логическая игра Сиджа

**Руководство программиста**

Р.0206933722/31217 РП-01

**Инв. № подл.**

**Подп. и дата**

**Взам. инв. №**

**Инв. № дубл.**

**Подп. и дата**

Листов 19

Исполнитель:

студент гр. ИСТбд-21

*Саранцев Евгений Андреевич*

« » 2023 г.

2023

**1. Назначение и условия применения программы**

**1.1 Назначение и функции, выполняемые приложением**

Функциональное назначение: разработка приложения для игры "Сиджа" с целью предоставления пользователю возможности играть в эту стратегическую настольную игру на своем устройстве.

Свод правил игры "Сиджа":

* Игра ведется на квадратной доске с нечетным количеством полей на стороне. Традиционная доска имеет размеры 5х5.
* Каждый из игроков получает количество камней равное половине полей доски за вычетом одного.
* Игра состоит из двух этапов. На первом этапе, игроки выставляют свои камни на поле.
* Запрещено выставлять камень только на центральное поле. Оно остается пустым до начала второй стадии.
* Вторую стадию игры начинает тот же игрок, который начинал и первую. На этой стадии, ход заключается в том, чтобы передвинуть камень на одно поле по горизонтали или вертикали.
* Результативным считается ход, после которого, хотя бы один камень противника окажется зажатым по горизонтали или вертикали или между камнем, выполнявшим ход и любым другим своим стоящим на доске.
* Камни противника, захваченные в соответствии с указанным правилом, снимаются с доски
* Если игроку удалось снять хотя бы один камень, то он получает право на второй ход.
* Игрок, взявший камень противника, имеет право выполнить ход любым своим камнем.
* Игрок не имеет право на дополнительный ход, а он обязан его выполнить, это обязательное правило.
* Игра завершается в двух случаях. Во-первых, если один из игроков потерял все свои камни. Во-вторых, если на протяжении десяти ходов не было выполнено ни одного взятия. В этом случае победителем считается игрок, сумевший взять больше камней противника.
* Захваченным считается камень противника, попавший между камнем игрока сделавшим ход и любым другим камнем игрока. При этом захваченными может считаться любое количество камней по любой горизонтали и вертикали.

Общая характеристика функциональных возможностей приложения:

Игра с компьютерным противником: приложение предоставляет возможность играть против компьютерного противника. Компьютерный противник может использовать различные алгоритмы и стратегии для принятия решений и выполнения ходов.

**1.2 Условия, необходимые для использования приложения**

Требования к операционной системе: Windows

Требования к платформе: настольный компьютер

**2. Характеристики программы**

**2.1 Характеристики приложения**

Количество значимых строк программного кода: 299.

Библиотеки, используемые при написании кода:

pygame - для разработки компьютерной игры.

sys - модуль, предоставляющий функциональность, специфичную для системы.

random - для генерации случайных ходов компьютера.

re - это модуль для работы с регулярными выражениями.

os - это модуль для работы с операционной системой.

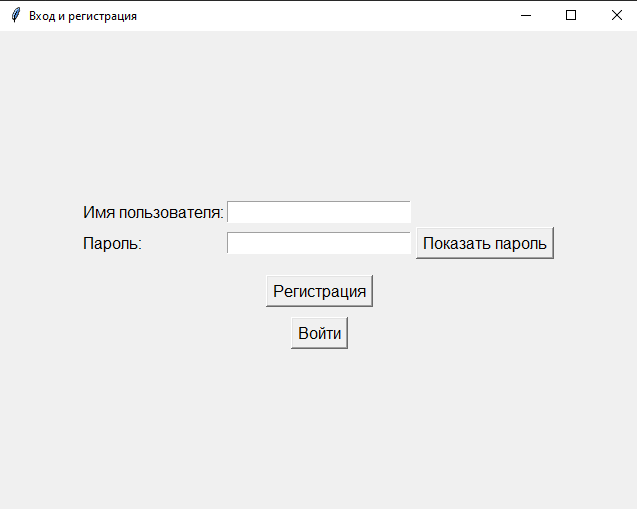
Tk - это класс из tkinter, который предоставляет основу для создания графического интерфейса с помощью библиотеки Tkinter.

Frame, Label, Entry, Button - это классы из tkinter для создания различных виджетов (компонентов) графического интерфейса.

messagebox - это модуль из tkinter для отображения диалоговых окон с сообщениями и вопросами.

tkfont - это модуль из tkinter для работы со шрифтами.

После запуска приложения у пользователя появляется окно входа/регистрации (рис.1).



*Рис. 1*

В появившемся окне необходим ввести своё имя пользователя. Если пользователь не зарегистрирован, ему необходимо придумать своё имя.  
Во втором поле необходимо ввести пароль, указанный при регистрации. Если аккаунт пользователя ещё не создан, то необходимо придумать пароль.

После успешной авторизации/регистрации появляется окно (рис.2).

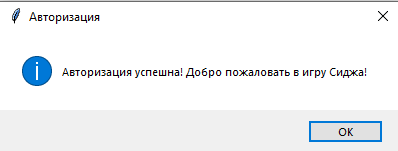
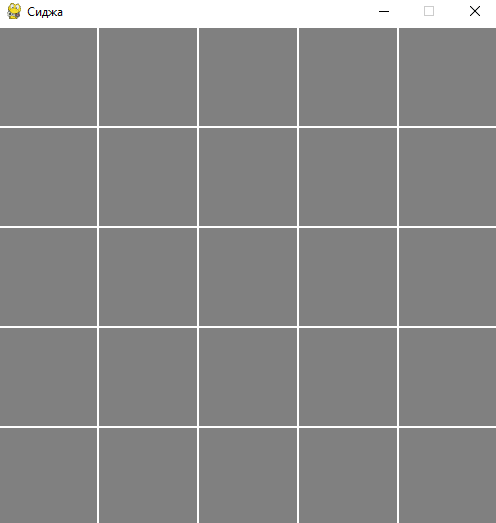


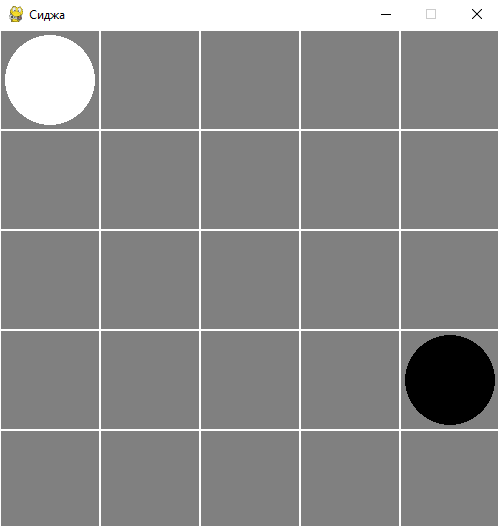
Рис.

После нажатия кнопки «ОК» появляется окно (рис. 3): на котором расположено игровое поле размеров 5x5 клеток.

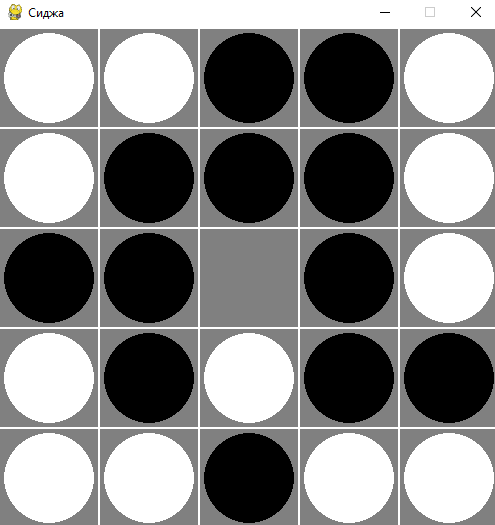


*Рис. 3*

Чтобы сделать ход, необходимо нажать на соответствующую клетку игрового поля. После хода пользователя, автоматически делает ход искусственный интеллект (рис. 4).

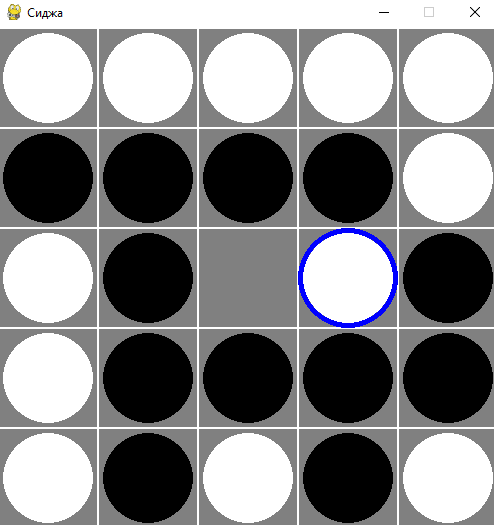
****

*Рис. 4*

Когда все клетки, кроме центральной, заполнены (Рис.5), начинается вторая фаза игры, где игрок должен выбрать фишку и передвинуть ее на свободную клетку.

*Рис. 5*

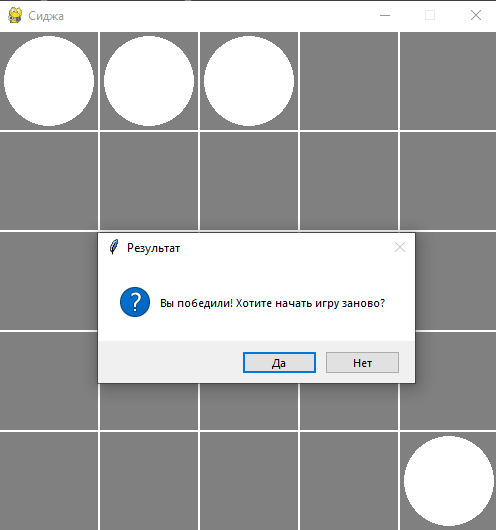
Чтобы сделать ход, необходимо нажать на соответствующую фишку. После хода пользователя, автоматически делает ход искусственный интеллект (Рис.6).



*Рис. 6*

Игра завершается в двух случаях. Во-первых, если один из игроков потерял все свои камни. Во-вторых, если на протяжении десяти ходов не было выполнено ни одного взятия. В этом случае победителем считается игрок, сумевший взять больше камней противника.

После завершения игры появляется окно «Результат», в котором написано победили вы, или проиграли. Так же, в этом окне, вам предлагается начать игру заново. (Рис.7).



*Рис. 7*

При нажатии на «Да», доска очищается, и игра полностью начинается заново (Рис.3).

При нажатии на «Нет», приложение через короткий промежуток времени самостоятельно закроется.

**2.2 Особенности реализации приложения**

Графический интерфейс: Приложение использует библиотеку Pygame для создания графического окна и отрисовки игрового поля. Окно имеет размер 500x500 пикселей, а каждая ячейка на игровом поле имеет размер 100x100 пикселей.

Игровое поле: Поле представляет собой квадрат размером 5x5. Он представлен в виде двумерного списка board, где каждый элемент может содержать значение "Игрок", "Компьютер" или None, в зависимости от того, занята ли ячейка и каким игроком.

Фазы игры: Игра состоит из двух фаз. В фазе 1 игрок и компьютер по очереди размещают свои фишки на свободных ячейках игрового поля. В фазе 2 они перемещают свои фишки по полю с целью захватить фишки противника.

**3. Обращение к программе**

Методы и алгоритмы

* Класс Game содержит различные методы для управления игрой.
* В конструкторе \_\_init\_\_ инициализируются основные параметры игры, создается игровое окно и инициализируется начальное состояние игровой доски.
* draw\_board отображает состояние игровой доски на экране. Каждая клетка доски представлена прямоугольником, а фишки игроков отображаются в виде окружностей разных цветов.
* any\_posible\_movs проверяет, есть ли у игрока возможные ходы.
* ai\_turn реализует ход компьютера в зависимости от текущей фазы игры. В фазе 1 компьютер выбирает случайную свободную клетку для хода. В фазе 2 компьютер перемещает свою фишку на соседнюю свободную клетку, если это возможно.
* check\_sidzha\_winner проверяет, есть ли победитель в игре. Если у одного из игроков нет возможных ходов, то другой игрок объявляется победителем.
* handle\_turn обрабатывает ход игрока. В фазе 1 игрок ставит свою фишку на свободную клетку. В фазе 2 игрок перемещает свою фишку на соседнюю свободную клетку, если это возможно.
* run запускает основной игровой цикл. Он обрабатывает события мыши и вызывает соответствующие методы для обработки ходов игроков. Также он отображает текущее состояние игры на экране и проверяет наличие победителя.
* Класс Authentication представляет систему аутентификации и регистрации пользователей для игры "Сиджа". Код использует библиотеки Tkinter и messagebox для создания графического интерфейса и взаимодействия с пользователем.
* В конструкторе \_\_init\_\_ инициализируются различные элементы графического интерфейса, такие как окно, поля ввода, кнопки и т.д. Они размещаются в окне с помощью различных методов Tkinter, таких как Frame, Label, Entry, Button и pack.
* reate\_database\_file создает файл базы данных, если он не существует. Файл используется для хранения информации о зарегистрированных пользователях.
* validate\_password проверяет, соответствует ли пароль заданному шаблону. В данной реализации пароль должен состоять только из цифр и латинских символов.
* is\_username\_available проверяет, доступно ли заданное имя пользователя для регистрации. Он читает файл базы данных и сравнивает каждую строку с заданным именем пользователя.
* register обрабатывает событие регистрации нового пользователя. Он получает введенное имя пользователя и пароль, а затем выполняет ряд проверок (например, наличие имени пользователя и пароля, длина пароля и доступность имени пользователя). Если все проверки проходят успешно, информация о новом пользователе добавляется в файл базы данных.
* login обрабатывает событие входа пользователя. Он получает введенное имя пользователя и пароль, а затем выполняет аналогичные проверки, как и в методе register, чтобы проверить правильность авторизационных данных. Если данные верны, отображается сообщение об успешной авторизации, а затем вызывается метод run класса Game для запуска игры "Сиджа".
* toggle\_password\_visibility переключает видимость символов пароля при нажатии на кнопку "Показать пароль". Если символы пароля скрыты, они отображаются, и наоборот.
* center\_window центрирует окно на экране, опираясь на заданные ширину и высоту окна. Он используется для установки положения окна аутентификации и регистрации по центру экрана.
* После создания экземпляра класса Authentication будет отображено окно аутентификации и регистрации, где пользователь может войти или зарегистрироваться для доступа к игре "Сиджа".

Библиотеки

* pygame: Это библиотека для разработки игр и мультимедийных приложений. Она предоставляет функции и классы для работы с графикой, звуком, анимацией и управлением игровым циклом.
* sys: Это модуль, предоставляющий функции и переменные, связанные с интерпретатором Python. В данном случае, он используется для выхода из программы с помощью функции sys.exit().
* random: Этот модуль предоставляет функции для генерации случайных чисел, выбора случайных элементов из последовательностей и других операций, связанных с случайностью.
* re: Это модуль для работы с регулярными выражениями в Python. Он предоставляет функции и методы для поиска и манипуляции текстом с использованием шаблонов.
* os: Это модуль, предоставляющий функции для взаимодействия с операционной системой. Он позволяет выполнять различные операции с файлами и директориями, управлять окружением и т.д.
* Tk из библиотеки tkinter: Это класс, представляющий основное окно приложения с графическим интерфейсом. Он используется для создания главного окна в приложении.
* Frame из библиотеки tkinter: Это класс, представляющий контейнер для других элементов интерфейса. Он используется для группировки элементов интерфейса и управления их расположением.
* Label из библиотеки tkinter: Это класс, представляющий текстовую метку в графическом интерфейсе. Он используется для отображения текста или изображения на экране.
* Entry из библиотеки tkinter: Это класс, представляющий поле ввода в графическом интерфейсе. Он используется для получения текстового ввода от пользователя.
* Button из библиотеки tkinter: Это класс, представляющий кнопку в графическом интерфейсе. Он используется для создания интерактивных элементов, на которые пользователь может нажимать.
* messagebox из библиотеки tkinter: Это модуль, предоставляющий функции для отображения диалоговых окон с сообщениями, предупреждениями и ошибками.
* tkfont из библиотеки tkinter: Это модуль, предоставляющий функции и классы для работы с шрифтами в графическом интерфейсе. Он используется для настройки внешнего вида текста в элементах интерфейса.

**4. Сообщения**

"Регистрация успешна!"

"Имя пользователя уже занято."

"Пароль должен состоять только из цифр и латинских символов."

"Пароль должен содержать минимум 8 символов."

"Введите имя пользователя и пароль."

"Авторизация успешна! Добро пожаловать в игру Сиджа!"

"Вы победили! Хотите начать игру заново?"

"Вы проиграли! Хотите начать игру заново?"

**Текст программы**

import pygame  
import sys  
import random  
import re  
import os  
from tkinter import Tk, Frame, Label, Entry, Button, messagebox  
from tkinter import font as tkfont  
  
class Game:  
 def \_\_init\_\_(self):  
 self.BOARD\_SIZE = 5  
 self.WINDOW\_SIZE = self.BOARD\_SIZE \* 100  
 pygame.init()  
 self.window = pygame.display.set\_mode((self.BOARD\_SIZE \* 100, self.BOARD\_SIZE \* 100))  
 pygame.display.set\_caption('Сиджа')  
 self.board = [[None for \_ in range(self.BOARD\_SIZE)] for \_ in range(self.BOARD\_SIZE)]  
 self.players = ['Игрок', 'Компьютер']  
 self.colors = {'Игрок': (255, 255, 255), 'Компьютер': (0, 0, 0)}  
 self.current\_player = self.players[0]  
 self.circles = {'Игрок': ((self.BOARD\_SIZE\*self.BOARD\_SIZE)-1)/2, 'Компьютер': ((self.BOARD\_SIZE\*self.BOARD\_SIZE)-1)/2}  
 self.game\_phase = 1  
 self.winner = None  
 self.lst\_psn = None  
  
 def draw\_board(self):  
 # Отрисовка игрового поля  
 self.window.fill((128, 128, 128))  
 for i in range(self.BOARD\_SIZE):  
 for j in range(self.BOARD\_SIZE):  
 pygame.draw.rect(self.window, (255, 255, 255), pygame.Rect(i \* 100, j \* 100, 100, 100), 1)  
 if self.board[i][j] is not None:  
 pygame.draw.circle(self.window, self.colors[self.board[i][j]], (i \* 100 + 50, j \* 100 + 50), 45)  
 if self.lst\_psn is not None and (i, j) == self.lst\_psn:  
 pygame.draw.circle(self.window, (0, 0, 255), (i \* 100 + 50, j \* 100 + 50), 50, 5)  
  
 def any\_posible\_movs(self, player):  
 # Проверка на возможность хода для игрока  
 for i in range(self.BOARD\_SIZE):  
 for j in range(self.BOARD\_SIZE):  
 if self.board[i][j] == player:  
 for di in [-1, 0, 1]:  
 for dj in [-1, 0, 1]:  
 if 0 <= i + di < self.BOARD\_SIZE and 0 <= j + dj < self.BOARD\_SIZE and self.board[i + di][j + dj] is None:  
 return True  
 return False  
  
 def ai\_turn(self, player):  
 # Ход ИИ  
 # Разграничение игроков для ИИ  
 directions = [(0, 1), (1, 0), (0, -1), (-1, 0)]  
  
 if self.game\_phase == 1:  
 empty\_cells = [(i, j) for i in range(self.BOARD\_SIZE) for j in range(self.BOARD\_SIZE) if  
 self.board[i][j] is None]  
 if empty\_cells:  
 i, j = random.choice(empty\_cells)  
 if (i, j) != (  
 self.BOARD\_SIZE // 2, self.BOARD\_SIZE // 2): # ИИ не занимает центральную клетку в первой фазе игры  
 self.handle\_turn(player, i, j)  
 elif self.game\_phase == 2:  
 player\_cells = [(i, j) for i in range(self.BOARD\_SIZE) for j in range(self.BOARD\_SIZE) if  
 self.board[i][j] == player]  
 random.shuffle(player\_cells)  
  
 for i, j in player\_cells:  
 for di, dj in directions:  
 if 0 <= i + di < self.BOARD\_SIZE and 0 <= j + dj < self.BOARD\_SIZE and self.board[i + di][j + dj] is None:  
 # Проверка, что ход приведет к захвату  
 if 0 <= i + 2 \* di < self.BOARD\_SIZE and 0 <= j + 2 \* dj < self.BOARD\_SIZE and \  
 self.board[i + 2 \* di][j + 2 \* dj] == player:  
 self.handle\_turn(player, i, j)  
 self.handle\_turn(player, i + di, j + dj)  
 # Удаление захваченной фишки  
 self.board[i + 2 \* di][j + 2 \* dj] = None  
 return  
  
 for i, j in player\_cells:  
 for di, dj in directions:  
 if 0 <= i + di < self.BOARD\_SIZE and 0 <= j + dj < self.BOARD\_SIZE and self.board[i + di][j + dj] is None:  
 self.handle\_turn(player, i, j)  
 self.handle\_turn(player, i + di, j + dj)  
 return  
  
 def check\_sidzha\_winner(self):  
 # Проверка на победителя в игре Сиджа  
 if self.game\_phase == 2:  
 if not self.any\_posible\_movs('Игрок'):  
 self.winner = 'Компьютер'  
 elif not self.any\_posible\_movs('Компьютер'):  
 self.winner = 'Игрок'  
  
 def handle\_turn(self, player, i, j):  
 # Обработка хода  
 opponent = self.players[0] if player == self.players[1] else self.players[1]  
 if self.game\_phase == 1:  
 if self.board[i][j] is None and (i, j) != (self.BOARD\_SIZE // 2, self.BOARD\_SIZE // 2):  
 self.board[i][j] = player  
 self.circles[player] -= 1  
 if self.circles['Игрок'] == 0 and self.circles['Компьютер'] == 0:  
 self.game\_phase = 2  
 # Проверка, кто окружил центральную клетку  
 for di, dj in [(-1, 0), (1, 0), (0, -1), (0, 1)]:  
 if self.board[self.BOARD\_SIZE // 2 + di][self.BOARD\_SIZE // 2 + dj] == player:  
 self.current\_player = player  
 break  
 elif self.board[self.BOARD\_SIZE // 2 + di][self.BOARD\_SIZE // 2 + dj] == opponent:  
 self.current\_player = opponent  
 break  
 else:  
 self.current\_player = opponent  
 self.lst\_psn = None  
 elif self.game\_phase == 2:  
 if self.lst\_psn is not None and self.board[i][j] is None:  
 old\_i, old\_j = self.lst\_psn  
 # Проверка, что фишка перемещается на одну клетку по горизонтали или вертикали  
 if abs(old\_i - i) + abs(old\_j - j) == 1:  
 self.board[old\_i][old\_j] = None  
 self.board[i][j] = player  
 for di, dj in [(-1, 0), (1, 0), (0, -1), (0, 1)]:  
 if 0 <= i + di < self.BOARD\_SIZE and 0 <= j + dj < self.BOARD\_SIZE:  
 if self.board[i + di][j + dj] == opponent and \  
 0 <= i + 2 \* di < self.BOARD\_SIZE and 0 <= j + 2 \* dj < self.BOARD\_SIZE and \  
 self.board[i + 2 \* di][  
 j + 2 \* dj] == player:  
 self.board[i + di][j + dj] = None  
 self.lst\_psn = None  
 self.check\_sidzha\_winner()  
 if self.winner is None:  
 self.current\_player = opponent  
 elif self.board[i][j] == player:  
 self.lst\_psn = (i, j)  
  
 def run(self):  
 while True:  
 for event in pygame.event.get():  
 if event.type == pygame.QUIT:  
 pygame.quit()  
 sys.exit()  
 elif event.type == pygame.MOUSEBUTTONDOWN and self.current\_player == self.players[0]:  
 x, y = pygame.mouse.get\_pos()  
 i, j = x // 100, y // 100  
 self.handle\_turn(self.current\_player, i, j)  
 elif self.current\_player == self.players[1]:  
 self.ai\_turn(self.current\_player)  
 self.window.fill((0, 0, 0))  
 self.draw\_board()  
 pygame.display.flip()  
 if self.winner is not None:  
 for i in range(self.BOARD\_SIZE):  
 for j in range(self.BOARD\_SIZE):  
 if self.board[i][j] == self.winner:  
 pygame.draw.circle(self.window, (0, 0, 255), (i \* 100 + 50, j \* 100 + 50), 50, 5)  
 if self.winner == "Игрок":  
 res = messagebox.askquestion("Результат", "Вы победили! Хотите начать игру заново?")  
 else:  
 res = messagebox.askquestion("Результат", "Вы проиграли! Хотите начать игру заново?")  
 if res == "yes":  
 self.board = [[None for \_ in range(self.BOARD\_SIZE)] for \_ in range(self.BOARD\_SIZE)]  
 self.players = ['Игрок', 'Компьютер']  
 self.colors = {'Игрок': (255, 0, 0), 'Компьютер': (0, 255, 0)}  
 self.current\_player = self.players[0]  
 self.circles = {'Игрок': ((self.BOARD\_SIZE \* self.BOARD\_SIZE) - 1) / 2,  
 'Компьютер': ((self.BOARD\_SIZE \* self.BOARD\_SIZE) - 1) / 2}  
 self.game\_phase = 1  
 self.winner = None  
 self.lst\_psn = None  
 else:  
 pygame.display.flip()  
 pygame.time.wait(5000)  
 pygame.quit()  
 sys.exit()  
  
class Authentication:  
 def \_\_init\_\_(self):  
 self.DATABASE\_FILE = "БД.txt"  
 self.show\_password = False  
 self.create\_database\_file()  
 self.root = Tk()  
 self.root.title("Вход и регистрация")  
 self.window\_width = 640  
 self.window\_height = 480  
 self.center\_window(self.root, self.window\_width, self.window\_height)  
  
 self.font = tkfont.Font(family="Arial", size=12)  
  
  
 self.main\_frame = Frame(self.root)  
 self.main\_frame.pack(expand=True)  
 self.frame = Frame(self.main\_frame)  
 self.frame.pack(pady=10,)  
  
 self.label\_username = Label(self.frame, text="Имя пользователя:", font=self.font)  
 self.label\_username.grid(row=0, column=0, pady=2, sticky="w")  
  
 self.label\_password = Label(self.frame, text="Пароль:", font=self.font)  
 self.label\_password.grid(row=1, column=0, pady=5, sticky="w")  
  
 self.entry\_username = Entry(self.frame, font=self.font)  
 self.entry\_username.grid(row=0, column=1)  
  
 self.entry\_password = Entry(self.frame, show="\*", font=self.font)  
 self.entry\_password.grid(row=1, column=1)  
  
 self.button\_show\_password = Button(self.frame, text="Показать пароль", font=self.font,  
 command=self.toggle\_password\_visibility)  
 self.button\_show\_password.grid(row=1, column=2, padx=5)  
  
 self.button\_register = Button(self.main\_frame, text="Регистрация", font=self.font, command=self.register)  
 self.button\_register.pack(pady=5)  
  
 self.button\_login = Button(self.main\_frame, text="Войти", font=self.font, command=self.login)  
 self.button\_login.pack(pady=5)  
  
 self.root.mainloop()  
  
 def create\_database\_file(self):  
 if not os.path.isfile(self.DATABASE\_FILE):  
 with open(self.DATABASE\_FILE, "w") as file:  
 pass  
  
 def validate\_password(self, password):  
 pattern = r'^[a-zA-Z0-9]+$'  
 return re.match(pattern, password) is not None  
  
 def is\_username\_available(self, username):  
 with open(self.DATABASE\_FILE, "r") as file:  
 lines = file.readlines()  
 for line in lines:  
 if "," in line:  
 stored\_username, \_ = line.strip().split(",")  
 if username == stored\_username:  
 return False  
 return True  
  
 def register(self):  
 username = self.entry\_username.get()  
 password = self.entry\_password.get()  
  
 if username and password:  
 if len(password) >= 8:  
 if self.validate\_password(password):  
 if self.is\_username\_available(username):  
 with open(self.DATABASE\_FILE, "a") as file:  
 file.write(f"{username},{password}\n")  
 messagebox.showinfo("Регистрация", "Регистрация успешна!")  
 else:  
 messagebox.showerror("Ошибка", "Имя пользователя уже занято.")  
 else:  
 messagebox.showerror("Ошибка", "Пароль должен состоять только из цифр и латинских символов.")  
 else:  
 messagebox.showerror("Ошибка", "Пароль должен содержать минимум 8 символов.")  
 else:  
 messagebox.showerror("Ошибка", "Введите имя пользователя и пароль.")  
  
 def login(self):  
 username = self.entry\_username.get()  
 password = self.entry\_password.get()  
  
 if username and password:  
 if len(password) >= 8:  
 if self.validate\_password(password):  
 with open(self.DATABASE\_FILE, "r") as file:  
 lines = file.readlines()  
 for line in lines:  
 if "," in line:  
 stored\_username, stored\_password = line.strip().split(",")  
 if username == stored\_username and password == stored\_password:  
 messagebox.showinfo("Авторизация", "Авторизация успешна! Добро пожаловать в игру Сиджа!")  
 self.root.withdraw() # Скрывает первое окно  
 Game().run()  
 return  
 messagebox.showerror("Ошибка", "Неверное имя пользователя или пароль.")  
 else:  
 messagebox.showerror("Ошибка", "Пароль должен состоять только из цифр и латинских символов.")  
 else:  
 messagebox.showerror("Ошибка", "Пароль должен содержать минимум 8 символов.")  
 else:  
 messagebox.showerror("Ошибка", "Введите имя пользователя и пароль.")  
  
 def toggle\_password\_visibility(self):  
 self.show\_password = not self.show\_password  
 if self.show\_password:  
 self.entry\_password.config(show="")  
 self.button\_show\_password.config(text="Скрыть пароль")  
 else:  
 self.entry\_password.config(show="\*")  
 self.button\_show\_password.config(text="Показать пароль")  
  
 @staticmethod  
 def center\_window(window, width, height):  
 screen\_width = window.winfo\_screenwidth()  
 screen\_height = window.winfo\_screenheight()  
 x = (screen\_width - width) // 2  
 y = (screen\_height - height) // 2  
 window.geometry(f"{width}x{height}+{x}+{y}")  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 Authentication()