**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 «Фризские шашки»

**Инв. № подл.**

**Подп. и дата**

**Взам. инв. №**

**Инв. № дубл.**

**Подп. и дата**

Р.02069337.22/2380-62

Листов 5

**Руководитель разработки**:

доцент каф. ИВК, к.т.н., доцент

*Шишкин Вадим Викторинович*

« » 2023 г.

**Исполнитель**:

студентка гр. ИСТбд-23

*Совалева Виктория Олеговна*

« » 2023 г.

**2023**

**Содержание**

Аннотация.......................................................................................1

Техническое задание.....................................................................3

Пояснительная записка ................................................................8

Руководство программиста ..........................................................13

Текст программы............................................................................25

**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 «Фризские шашки»

**Инв. № подл.**

**Подп. и дата**

**Взам. инв. №**

**Инв. № дубл.**

**Подп. и дата**

Р.02069337.22/2380-62 ТЗ-03

Листов 5

**Исполнитель**:

студентка гр. ИСТбд-23

*Совалева Виктория Олеговна*

« » 2023 г.

**2023**

**Введение**

Компьютерная логическая игра «Фризские шашки»

Приложение должно соответствовать правилам игры, приведенным ниже.

**Поле и игроки.** Играют двое игроков. Шашечная доска состоит из 100 (10х10) одинаковых клеток, попеременно светлых и темных. Шашки расставляются на чёрных полях первых четырёх горизонтальных рядов с каждой стороны. Пользователь ходит белыми первый и делает первых ход. Далее ходы делаются поочередно.

**Шашки.** Шашки подразделяются на простые и дамки. Дамки маркируются знаком короны. При достижении любого поля последней горизонтали, простая шашка превращается в дамку. Партнерам перед началом игры предоставляется по 20 простых шашек: одному светлых, другому — темных.

**Стартовая позиция.** Шашки расставляются на темных клетках первых четырех горизонтальных рядов с каждой стороны.

**Ходы.** Ходом в партии считается передвижение шашки с одной клетки поля на другое. Первый ход всегда делает игрок, играющий светлыми. Игроки поочередно делают по одному ходу до тех пор, пока игра не закончится. Шашки разделяются на простые шашки и дамки. Если шашка попадет на клетку, где расположена шашка противника, эта шашка считается «съеденной» и убирается с доски. Простая шашка ходит по диагонали вперед на одну клетку. При достижении любого поля последней горизонтали, простая шашка превращается в дамку. Дамка обозначаетсясимволом короны. Дамка, в отличие от простой шашки, ходит на любое из свободных (не занятых другими шашками) полей по диагонали в любом направлении (как вперед, так и назад), но становиться может, как и простая шашка, лишь на не занятые другими шашками клетки, причем через свои шашки она перескакивать не может.

**Взятие.** Если простая шашка находится на одной диагонали, горизонтали или вертикали рядом с шашкой другого игрока, за которой имеется свободная клетка, она должна быть перенесена через эту шашку на свободную клетку. При взятии по горизонтали или вертикали простая шашка становится на следующую свободную черную клетку. Шашка другого игрока в этом случае снимается с поля. Если дамка находится на одной диагонали рядом или на расстоянии от шашки другого игрока, за которой имеется одно или несколько свободных клеток, она должна быть перенесена через эту шашку на любую свободную клетку. Шашка другого игрока в этом случае снимается с поля. Взятие обязательно. Направления взятия равноправны. При взятии применяется правило турецкого удара — если при бое нескольких шашек противника шашка или дамка повторно выходит на уже побитую шашку, то ход останавливается (то есть, запрещается дважды брать одну и ту же шашку, при этом можно пересекать дважды одно и то же пустое поле). Если есть несколько вариантов взятия, нужно выполнить тот из них, при котором бьётся максимальное количество шашек соперника. При этом, дамка считается ценнее простой шашки, то есть при возможности взятия, например, двух простых или простой и дамки необходимо бить простую и дамку. Ценность двух простых шашек выше, чем ценность одной дамки. Если имеется выбор боя дамкой или простой одинакового количества шашек, то необходимо бить дамкой. В случае, когда может быть взято большее количество дамок, действует следующее правило: число дамок имеет ценность вдвое большего числа шашек минус ½. То есть, три дамки, которые могут быть побиты, имеют ценность 5½ простых шашек. При достижении в эндшпиле положения двух дамок против одной, соперник владеющий двумя дамками должен взять дамку соперника в течение 7 ходов. Если ему это не удаётся, то партия признаётся ничейной.

### **Выигрыш партии**. Партия считается выигранной в следующих случаях: 1) если у одного из соперников побиты все шашки 2) если шашки одного из участников заперты, и он не может сделать очередной ход.

**1. Основания для разработки**

Учебный план направления 09.03.02 «Информационные системы и технологии» и распоряжение по факультету «О закреплении тем курсовых работ (проектов) за студентами 2 курса ФИСТ направления 09.03.02 «Информационные системы и технологии» (профиль Информационные системы и технологии) по дисциплине «Алгоритмы и структуры данных» от 20.10.2023

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

**2.1. Функциональное назначение**

Требуется разработать десктопное приложение для двух пользователей по игре в шашки с графическим интерфейсом в среде Windows.

Перечень автоматизируемых процессов:

1. Регистрация и авторизация пользователей
2. Отрисовка игрового поля
3. Обработка событий мыши
4. Проверка правильности ходов
5. Проверка на конец игры

**2.2 Требования к функциональным характеристикам**

2.2.1 Требования к структуре приложения

Общая структура приложения должна быть разделена на 2 компонента.

Первый отвечает за авторизацию/регистрацию, за личный кабинет и переход к игре.

Второй отвечает за саму игру, ее отрисовку, проверку на правила и на конец игры.

2.2.2 Требования к составу функций приложения

В приложении должны быть реализованы в графическом режиме следующие основные функции:

- регистрация/авторизация пользователя;

- отрисовка игрового поля;

- взаимодействие с пользователем;

- интерактивные прием, проверка правильности и отрисовка хода пользователя;

- проверка окончания игры;

- вычисление, проверка правильности и отрисовка хода компьютера.

- информирование пользователя об окончании игры и победителе.

2.2.2 Требования к организации информационного обеспечения, входных и выходных данных

В приложении должен быть реализован графический интерфейс взаимодействия с пользователем. Отдельно выделены папки под графические файлы, шрифт, заготовку объектов и карты, аудио эффектов, а также для самого кода. Логин и пароль пользователя должны вводиться с клавиатуры. Логины и пароли зарегистрированных пользователей должны храниться в отдельном файле или базе данных в зашифрованном виде.

**2.3 Требования к надёжности**

Поддержка непрерывной и стабильной работы компьютера.

**2.4 Требования к информационной и программной совместимости**

Рекомендуется к использованию на Windows 10/11.

При создании программы используются встроенные библиотеки “random”, “os”, “tkinter”.

Разработка ведётся в “PyCharm community edition 2023.2.3” на версии языка программирования Python 3.11.5.

**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

Диск CD-R должен храниться при комнатной температуре, в диапазоне от 20°C до 25°C. Рекомендуется хранить диск в условиях с относительной влажностью воздуха от 20% до 50%. Диск CD-R должен храниться в темном месте, защищенном от прямых солнечных лучей и других источников яркого света. Для предотвращения повреждения диска CD-R рекомендуется хранить его в специальных пластиковых коробках или футлярах, предназначенных для хранения CD-дисков.

2.6 3 Сроки хранения

Срок хранения – до июля 2024 года**.**

**3. Требования к программной документации**

Определяется заданием на курсовую работу.

**4. Стадии и этапы разработки**

Определяется заданием на курсовую работу.

**5. Порядок контроля и приёмки**

Определяется 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 «Фризские шашки»

Пояснительная записка

**Инв. № подл.**

**Подп. и дата**

**Взам. инв. №**

**Инв. № дубл.**

**Подп. и дата**

Р.02069337.22/2380-62 ПЗ-03

Листов 5

**Руководитель разработки**:

доцент каф. ИВК, к.т.н., доцент

*Шишкин Вадим Викторинович*

« » 2023 г.

**Исполнитель**:

студент гр. ИСТбд-23

*Совалева Виктория Олеговна*

« » 2023 г.

**2023**

**Введение**

Приложение «Фризские шашки» реализует функционал логической игры «Фризские шашки»

В качестве подхода для разработки была выбрана каскадная модель разработки («Водопад»). Каскадная модель была выбрана из-за своей простоты, она позволяет наглядно представить объём работы и сроки выполнения, а также эффективно разбить проект на несколько подзадач.

Приложение «Фризские шашки» представляет собой игру Фризские шашки на 100 клеточном поле (10 клеток в ширину и 10 в длину) для двоих игроков.

**1. Проектная часть**

**1.1 Постановка задачи на разработку приложения**

Определяется заданием на курсовую работу

**1.2 Математические методы**

В качестве математической модели для представления и отрисовки поля был выбран двумерный массив, он позволяет легко записать положение всех шашек, выполнять все необходимые проверки и отрисовывать шашки, а также может быть легко изменен, что упрощает вывод хода пользователя. В массиве пустые клетки представлены числом 0, белые шашки 1, белые дамки 2, черные шашки -1, черные дамки -2.

**1.3 Архитектура и алгоритмы**

1.3.1. Архитектура



1. Регистрация/Авторизация.

Интерфейс, созданный с помощью библиотеки Tkinter позволяющий пользователю зарегистрировать новый аккаунт и сыграть с помощью уже существующего.

2. Отрисовка основных виджетов.

Вывод на экран игрового поля с расставленными шашками

3. Обработчик событий.

Алгоритм проверки правильности хода пользователя, в случае подтверждения ввода пользователем правильного хода, позволяет в дальнейшем вывести его на экран.

4. Отрисовка ходов.

Вывод на экран ходов пользователей.

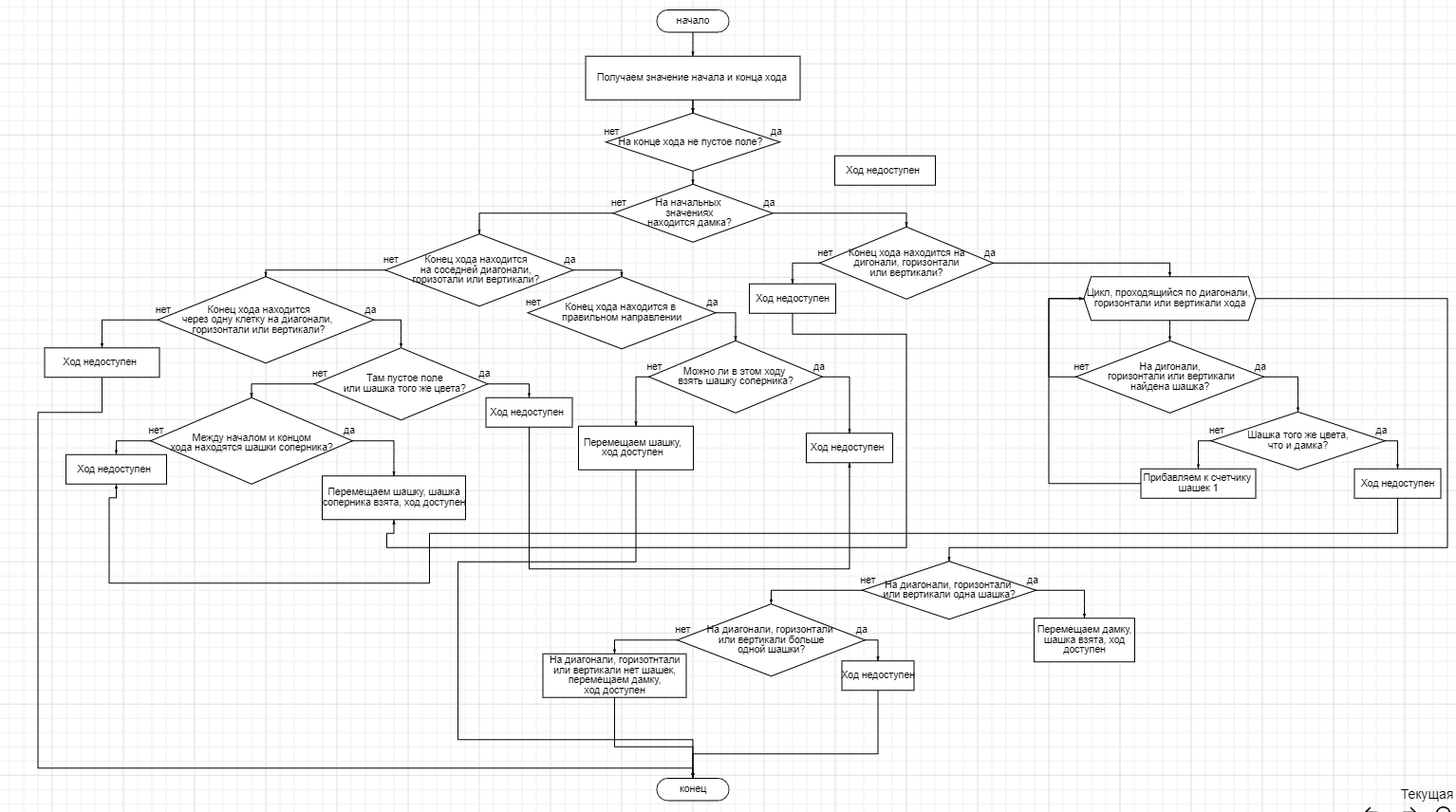
5. Определение конца игры.

Алгоритм проверки окончания игры, проверяет после хода пользователя закончена ли игра, в случае конца игры выводит на экран победителя.

1.3.2.1. Алгоритм проверки регистрации

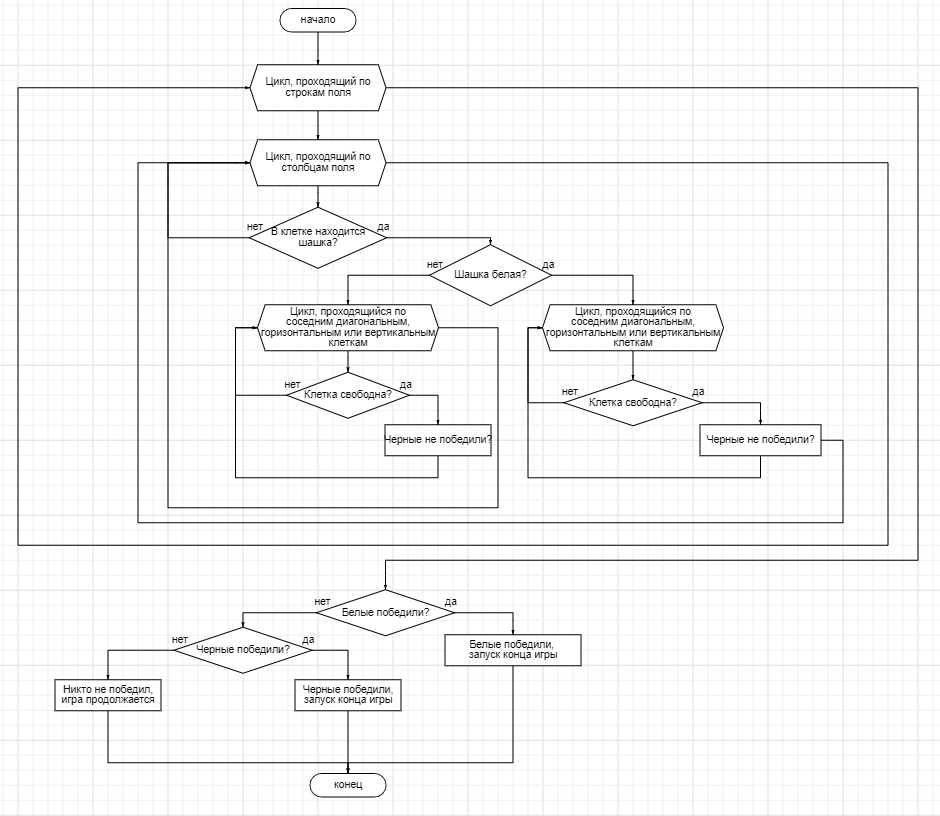
Данный алгоритм необходим для проверки регистрации пользователя. Алгоритм принимает логин и пароль, шифрует пароль, сверяет полученные данные с данными о зарегистрированных пользователях. Если пользователь с таким логином найден и пароль соответствует записанному в файл с зарегистрированными, то алгоритм позволяет начать игру, иначе выводит предупреждение о том, что данные некорректны.

1.3.2.2. Алгоритм проверки доступности хода.



Данный алгоритм необходим для контроля за соблюдением пользователем правил игры. Алгоритм проверяет возможен ли ход, который выбрал пользователь. Алгоритм получает на вход два индекса двумерного массива. Первый индекс указывает на шашку, которой пользователь хочет сходить, а второй на клетку, в которую он хочет сходить. Алгоритм проверяет возможность такого хода и, если он возможен двигает шашку пользователя, иначе ничего не делает и пользователь должен заново выбрать поле для хода.

1.3.2.3. Алгоритм проверки конца игры.



Данный алгоритм необходим для того, чтобы определить победителя и корректно закончить игру. Алгоритм проверяет наличие шашек и ходов и в случае их отсутствия запускает сценарий конца игры, иначе ничего не делает и игра продолжается.

**1.4 Тестирование**

Весь процесс тестирования проходил вручную, без привлечения специального ПО. На протяжении всего хода разработки, использовался метод белого ящика, так как в любом время имелся доступ ко всем компонентам программы. Всё тестирование выполнялось интуитивным методом, без подготовки специальных тестов.

На протяжении всего хода разработки, по мере добавления новых функций программы, использовалось системное тестирование новых функций, для устранения возникших в ходе написания ошибок. После положительных результатов тестирования функция считалась внедренной.

**2. Источники, использованные при разработке**

Разработка логических компьютерных игр с графическим интерфейсом в среде Питон: учебное пособие для студентов направления 09.03.02 «Информационные системы и технологии» / В.В. Шишкин, Д.С. Афонин. – Ульяновск : УлГТУ, 2023. – 88 с.

**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 «Фризские шашки»

Руководство программиста

**Инв. № подл.**

**Подп. и дата**

**Взам. инв. №**

**Инв. № дубл.**

**Подп. и дата**

Р.02069337.22/2380-62 РП-01

Листов 14

**Исполнитель**:

студент гр. ИСТбд-23

*Совалева Виктория Олеговна*

« » 2023 г.

**2023**

**1. Назначение и условия применения программы**

**1.1 Назначение и функции, выполняемые приложением**

Приложение «Фризские шашки» реализовано для развлечения пользователей. Приложение реализовано по следующим правилам:

**Поле и игроки.** Играют двое игроков. Шашечная доска состоит из 100 (10х10) одинаковых клеток, попеременно светлых и темных. Шашки расставляются на чёрных полях первых четырёх горизонтальных рядов с каждой стороны. Пользователь ходит белыми первый и делает первых ход. Далее ходы делаются поочередно.

**Шашки.** Шашки подразделяются на простые и дамки. Дамки маркируются знаком короны. При достижении любого поля последней горизонтали, простая шашка превращается в дамку. Партнерам перед началом игры предоставляется по 20 простых шашек: одному светлых, другому — темных.

**Стартовая позиция.** Шашки расставляются на темных клетках первых четырех горизонтальных рядов с каждой стороны.

**Ходы.** Ходом в партии считается передвижение шашки с одной клетки поля на другое. Первый ход всегда делает игрок, играющий светлыми. Игроки поочередно делают по одному ходу до тех пор, пока игра не закончится. Шашки разделяются на простые шашки и дамки. Если шашка попадет на клетку, где расположена шашка противника, эта шашка считается «съеденной» и убирается с доски. Простая шашка ходит по диагонали вперед на одну клетку. При достижении любого поля последней горизонтали, простая шашка превращается в дамку. Дамка обозначаетсясимволом короны. Дамка, в отличие от простой шашки, ходит на любое из свободных (не занятых другими шашками) полей по диагонали в любом направлении (как вперед, так и назад), но становиться может, как и простая шашка, лишь на не занятые другими шашками клетки, причем через свои шашки она перескакивать не может.

**Взятие.** Если простая шашка находится на одной диагонали, горизонтали или вертикали рядом с шашкой другого игрока, за которой имеется свободная клетка, она должна быть перенесена через эту шашку на свободную клетку. При взятии по горизонтали или вертикали простая шашка становится на следующую свободную черную клетку. Шашка другого игрока в этом случае снимается с поля. Если дамка находится на одной диагонали рядом или на расстоянии от шашки другого игрока, за которой имеется одно или несколько свободных клеток, она должна быть перенесена через эту шашку на любую свободную клетку. Шашка другого игрока в этом случае снимается с поля. Взятие обязательно. Направления взятия равноправны. При взятии применяется правило турецкого удара — если при бое нескольких шашек противника шашка или дамка повторно выходит на уже побитую шашку, то ход останавливается (то есть, запрещается дважды брать одну и ту же шашку, при этом можно пересекать дважды одно и то же пустое поле). Если есть несколько вариантов взятия, нужно выполнить тот из них, при котором бьётся максимальное количество шашек соперника. При этом, дамка считается ценнее простой шашки, то есть при возможности взятия, например, двух простых или простой и дамки необходимо бить простую и дамку. Ценность двух простых шашек выше, чем ценность одной дамки. Если имеется выбор боя дамкой или простой одинакового количества шашек, то необходимо бить дамкой. В случае, когда может быть взято большее количество дамок, действует следующее правило: число дамок имеет ценность вдвое большего числа шашек минус ½. То есть, три дамки, которые могут быть побиты, имеют ценность 5½ простых шашек. При достижении в эндшпиле положения двух дамок против одной, соперник владеющий двумя дамками должен взять дамку соперника в течение 7 ходов. Если ему это не удаётся, то партия признаётся ничейной.

### **Выигрыш партии**. Партия считается выигранной в следующих случаях: 1) если у одного из соперников побиты все шашки 2) если шашки одного из участников заперты, и он не может сделать очередной ход.

В приложении предоставляется возможность зарегистрироваться чтобы играть со своим аккаунтом, сыграть в логическую игру Фризские шакши, в приложении реализованы все правила этой игры: порядок хода игроков, ходы шашек, взятие шашки, превращения шашки в дамку, обязательное взятие шашки, последовательное взятие, проверка конца игры и вывод победителя.

**1.2 Условия, необходимые для использования приложения**

Рекомендуется к использованию на Windows 10/11.

При создании программы использовались встроенная библиотека “os”, “random” и сторонняя библиотека “tkinter”.

Разработка велась в “PyCharm community edition 2023.2.3” на версии языка программирования Python 3.11.5.

**2. Характеристики программы**

**2.1 Характеристики приложения**

Значимых строк кода 564.

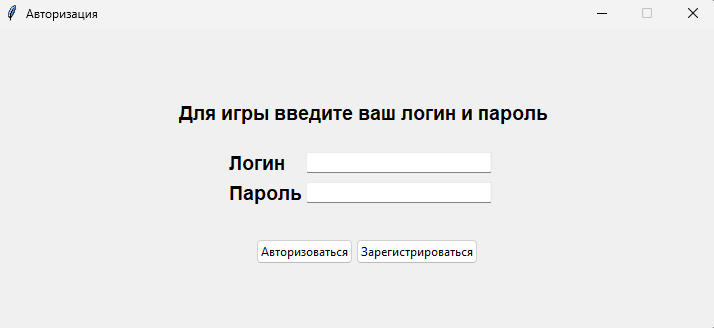
Структура данных одна – массив.

Использованные библиотеки:

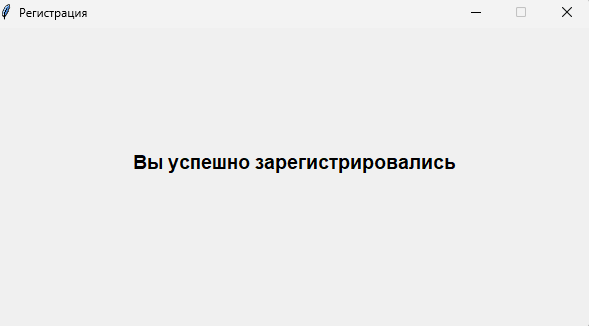
“os” –библиотека функций для работы с операционной системой. Методы, включенные в неё, позволяют определять тип операционной системы, получать доступ к переменным окружения, управлять директориями и файлами

“tkinter” - библиотека для разработки графического интерфейса на языке Python. Методы, включенные в неё, позволяют создавать окна, размещать на них виджеты, настраивать параметры окна и виджетов.

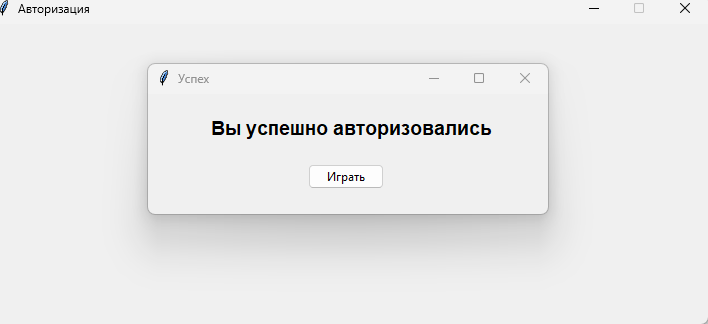
Работа приложения начинается с окна авторизации и регистрации



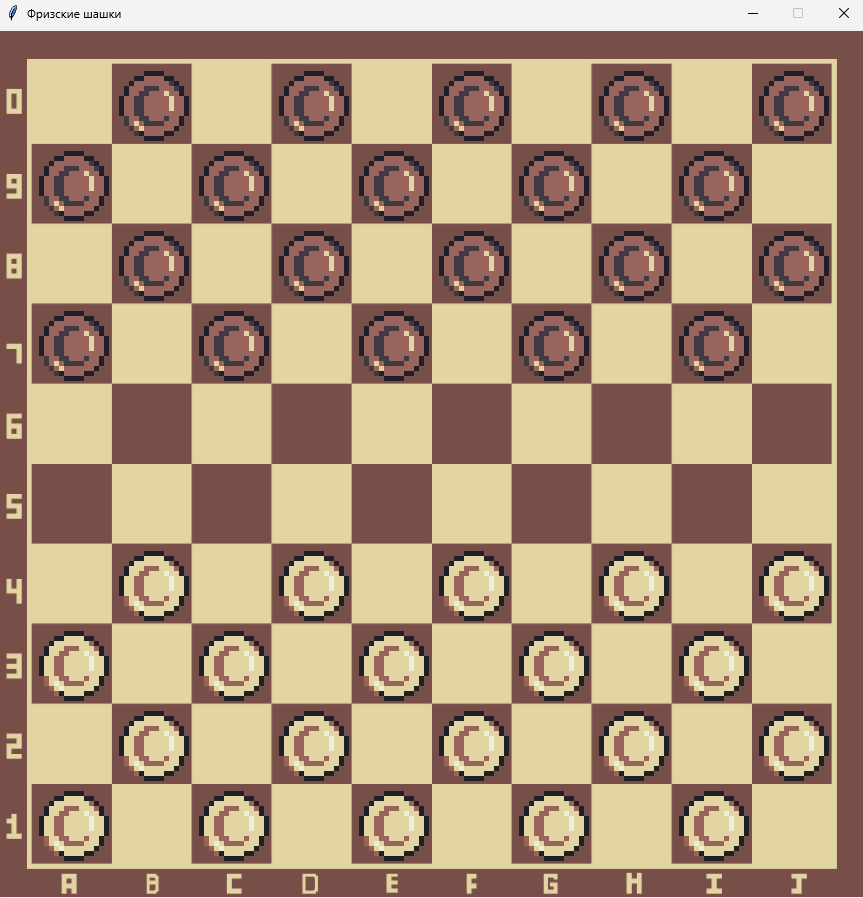
Успешная регистрация



Успешная авторизация



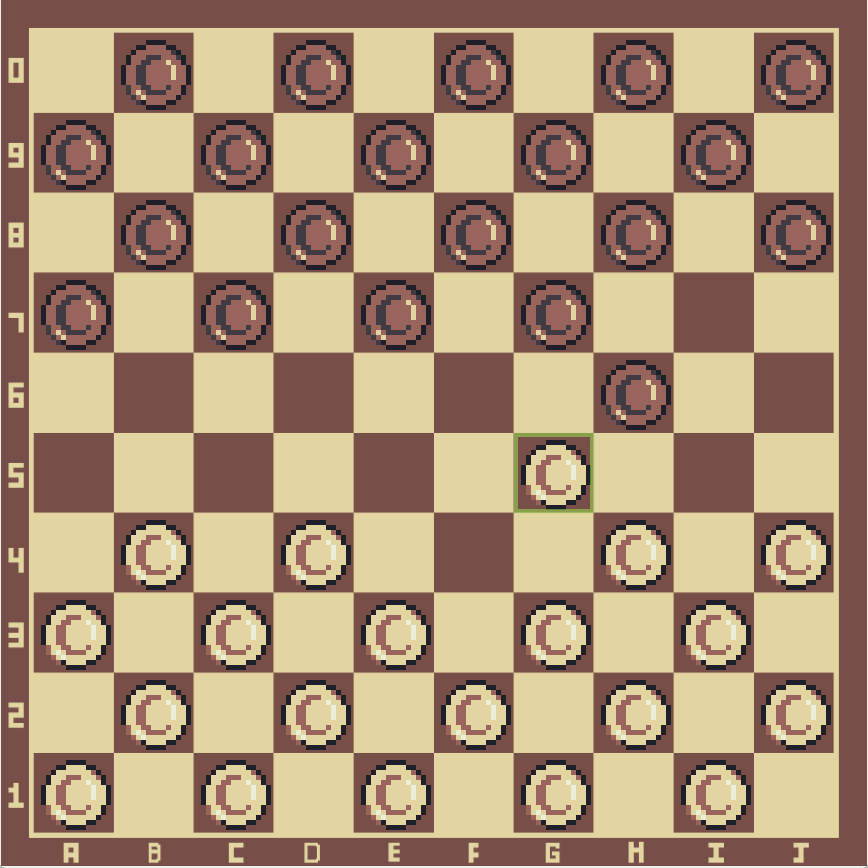
Отрисовка поля

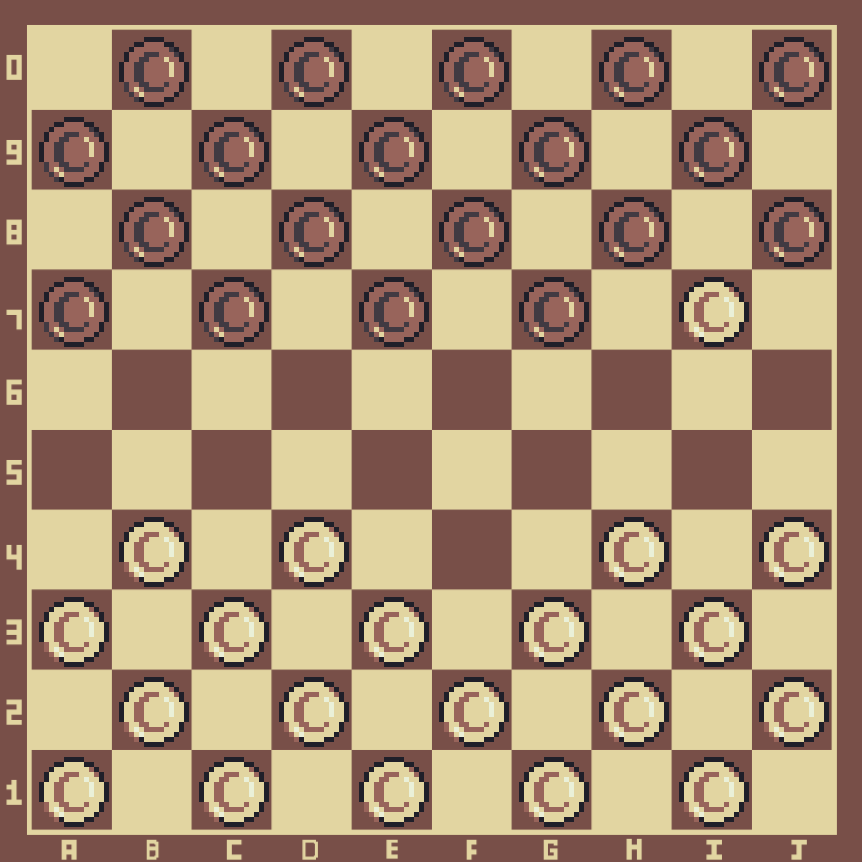


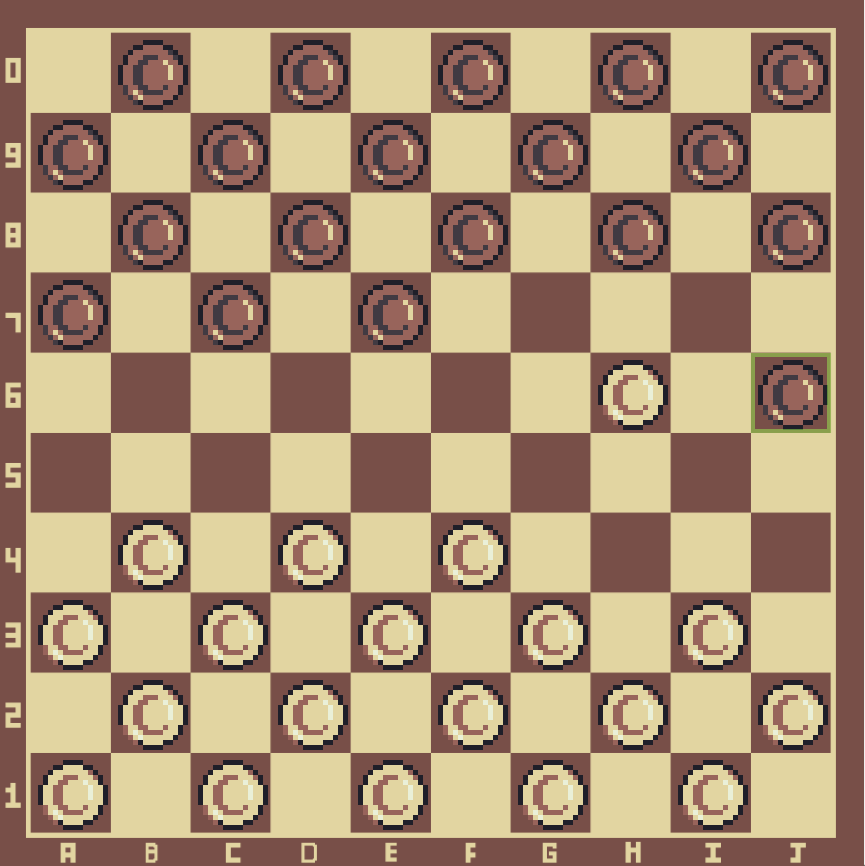
Выбор шашки для хода

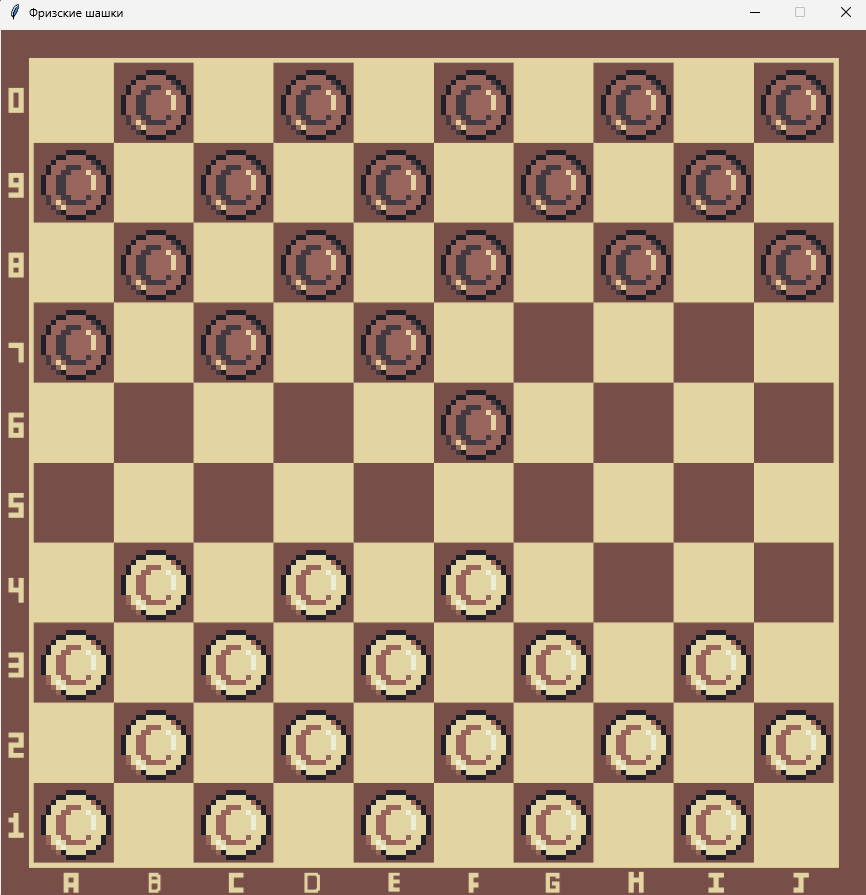


Взятие шашки

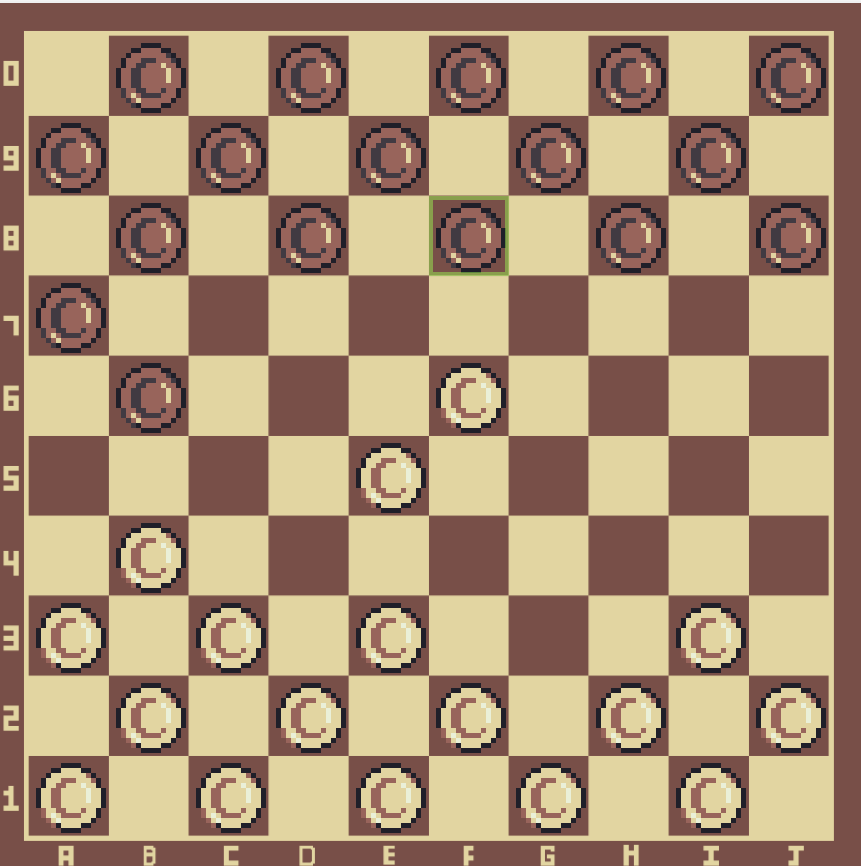
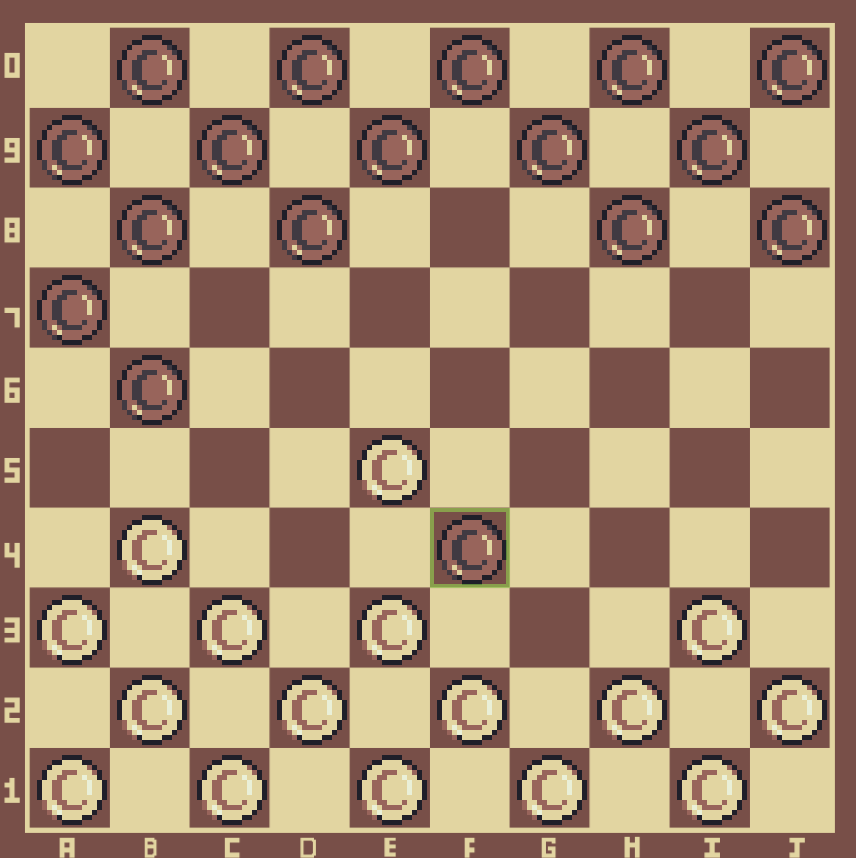


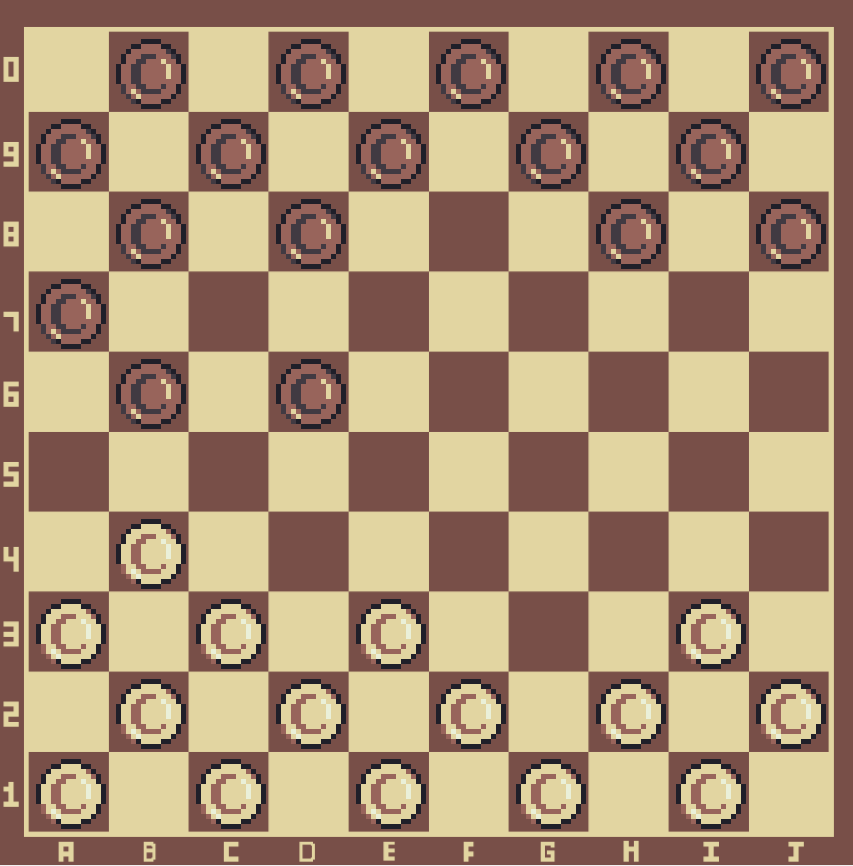




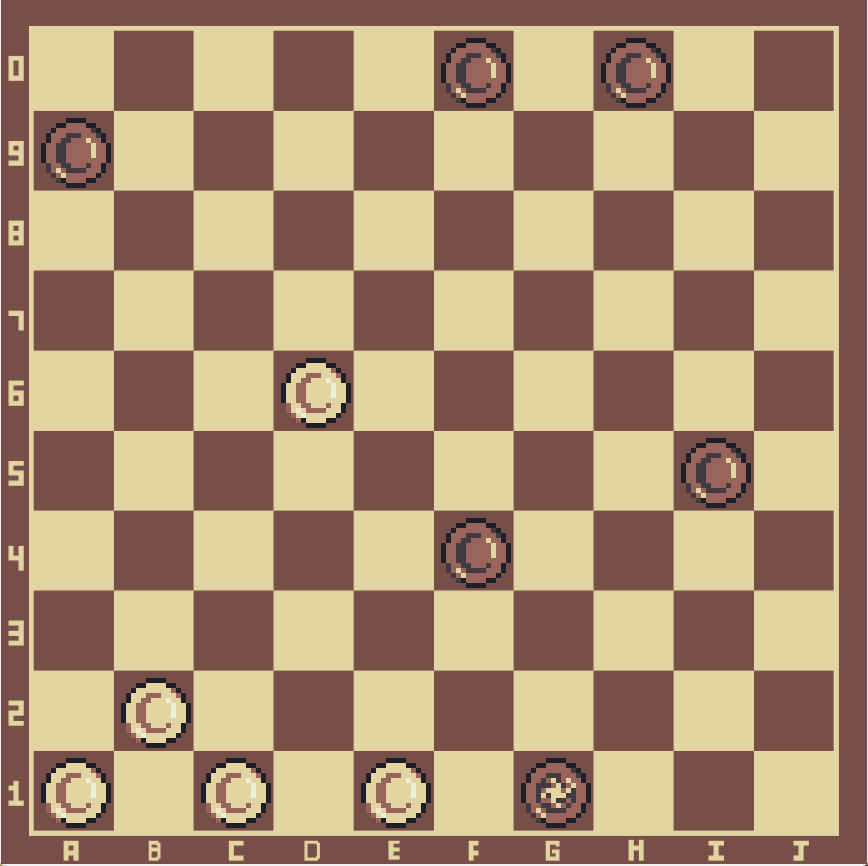


Последовательное взятие

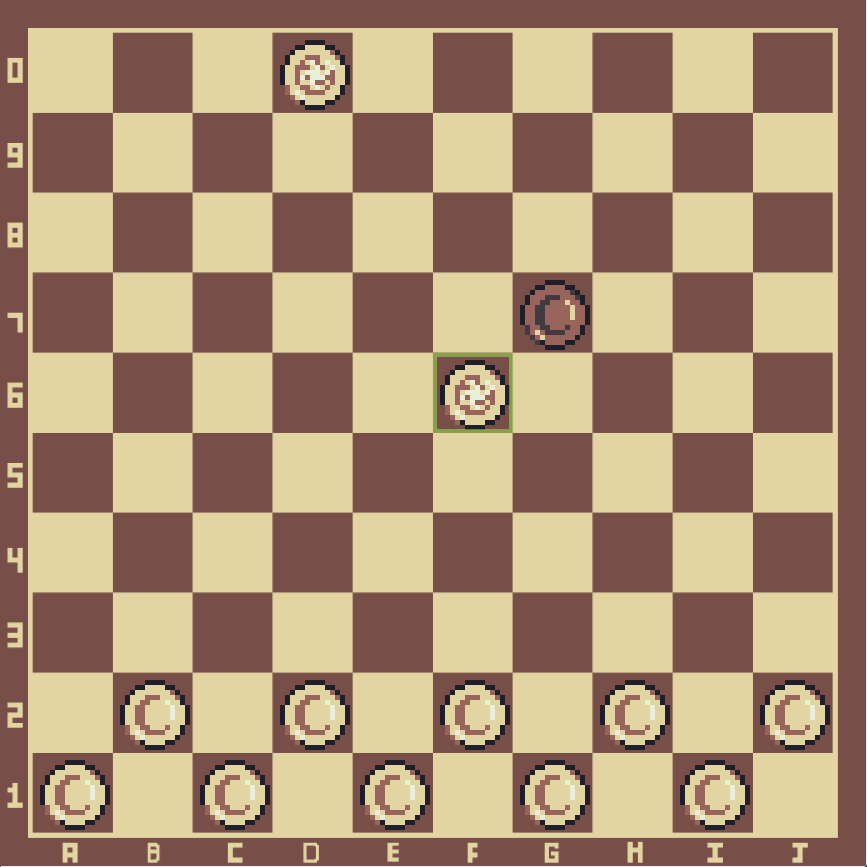




Превращение в дамку



Окончание игры и возможность перезапуска



**2.2 Особенности реализации приложения**

В программе используется структура данных массив, эта структура была выбрана ввиду удобной навигации для отрисовки и проверок хода, обязательного взятия, превращения шашки в дамку и последовательного боя, т.к. двумерном массиве есть возможность поиска по строке и столбцу, что приближено к реальному шашечному полю, у которого также есть строки и столбцы

**3. Обращение к программе**

Класс Checkers содержит в себе следующие методы:

1) Метод authorization

Авторизация пользователя, проверяет наличие логина и соответствующего ему пароля в файле с записанными данными пользователей

2) Метод regist

Регистрация пользователя, проверяет наличие логина в файле с записанными данными пользователей и если пользователь с таким логином ещё не зарегистрирован, то вносит данные о пользователе.

3) Метод drawing\_field

Отрисовка доски, отрисовка шашек в соответствии с данными из массива для представления поля

4) Метод drawing\_move

Отрисовка хода пользователя, удаляет все элементы из окна и заново отрисовывает доску и новое расположение шашек на окне, из-за высокой скорости отрисовки всё происходит незаметно для пользователя.

5) Метод interact

Обработчик событий, получает данные о нажатии пользователя в окне игры, с помощью алгоритмов проверки хода, проверки обязательного взятия, последовательного боя и превращения шашки в дамку реализует действия на окне игры

6) Метод checkers\_end

Проверяет наличие ходов для каждой стороны

7) Метод end\_game

Выводит на экран данные о победителе, блокирует нажатия по полю игры и позволяет начать новую игру

Алгоритмы, используемые в программе:

1) password\_code

Шифрование пароля пользователя, сдвигает каждый символ пароля с помощью ключа и возвращает зашифрованный пароль

2) open

Открывает файл с данными о регистрации пользователей, если не находит его, то создаёт новый файл

3) dismiss

Получает окно и позволяет перехватить управление

4) queen\_check

Проверяет, находится ли шашка на нужном поле для превращения в дамку, если да, то меняет её представление в модели поля

5) attack\_check

Получает данные о ходе пользователя. Проходит циклами по всей длине модели поля, если находит шашку или дамку, то проверяет наличие у неё возможности забрать шашку или дамку другого цвета. Для обычной шашки проходит циклом по соседним клеткам, для дамки проходит циклом по всем диагоналям. Если находит возможность взятия, то записывает их в массив. Если данные о ходе пользователя совпадают с записанными во время проверки возможности взятия, то позволяет сделать ход. Если не находит возможность взятия, то позволяет сделать ход.

6)move\_check

Получает данные о ходе пользователя. Проверяет на конечных координатах хода наличие пустого поля, если поле не пустое, то не позволяет сделать ход, иначе проверяет наличие на начальных координатах дамки, если на начальных координатах дамка, то проходит циклом по диагонали хода и если находит там шашку или дамку своего цвета, то не позволяет сделать ход, иначе прибавляет к счётчику шашек 1, если после завершения цикла счётчик шашек равен 1, то позволяет сделать ход и меняет представление шашки другого цвета в массиве на пустое поле, если счётчик шашек больше 1, то не позволяет сделать ход, если равен 0, то проверяет с помощью алгоритма обязательного взятия(attack\_check) возможность взятия шашки или дамки, если не находит возможности взятия, то позволяет сделать ход, иначе не позволяет.

Если на начальных координатах не дамка, то проверяет на сколько клеток сделан ход и находится ли он на диагонали, горизонтали или вертикали, если ход сделан на 2 клетки и по диагонали, горизонтали или вертикали, то проверяет наличие на среднем значении между координатами начала и конца хода шашки или дамки другого цвета, если находит, то позволяет сделать ход, иначе не позволяет, если ход сделан на одну клетку то проверяет в правильном ли он направлении, если да проверяет с помощью алгоритма обязательного взятия(attack\_check) возможность взятия шашки или дамки, если не находит возможности взятия, то позволяет сделать ход, иначе не позволяет

7)multijump\_check

Получает данные о только что перемещённой шашке или дамке. Проверяет имеется ли возможность у только что перемещённой шашки или дамки продолжать бой. Для обычной шашки проходит циклом по соседним клеткам, для дамки проходит циклом по всем диагоналям. Если находит возможность взятия, то позволяет совершать последовательный бой.

8) click\_event

Получает данные о клике пользователя и преобразует их в координаты начала хода. Проверяет очередность хода и находится ли на координатах начала шашка или дамка нужного цвета, если да, то выделяет её и позволяет выбрать координаты конца хода.

9) click\_event\_capt

Получает данные о клике пользователя и преобразует их в координаты конца пользователя. Если в этом ходу ещё не было перемещений (переменная self.first\_move) , то позволяет отменить ход нажатием на шашку, иначе не позволяет отменить ход и обязует пользователя продолжать бой. Проверяет с помощью алгоритмов проверки правильности хода (move\_check) корректность хода, если ход корректен, то вызывает метод отрисовки хода(drawing\_move), проверяет с помощью алгоритма проверки последовательного взятия(multijump\_check) возможность последовательного взятия, если не находит, то передаёт ход другой стороне и передаёт управление алгоритму получения данных о начальных координатах (click\_event), иначе меняет переменную первого хода(self.first\_move) и обязует пользователя продолжать взятие. Если обнаруживает, что в этом ходу уже было взятие, то блокирует значение начальных координат и проверяет с помощью алгоритма проверки правильности хода(move\_check) корректность боя, если находит возможность дальнейшего взятия, то повторяет, иначе передаёт ход другой стороне и передаёт управление алгоритму получения данных о начальных координатах (click\_event).

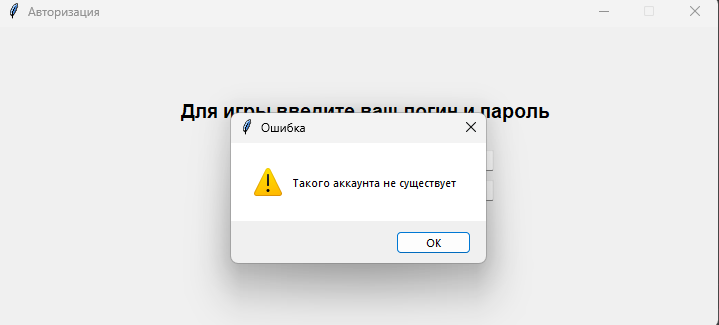
Библиотеки, используемые во время разработки приложения:

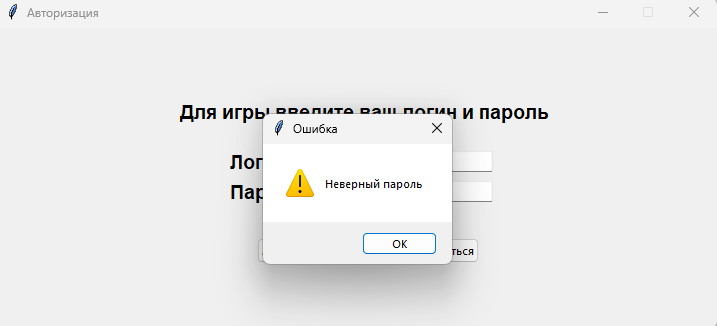
Библиотека “os” - предоставляет возможность взаимодействовать с операционной системой, независимо от того, какая из них установлена на устройстве, что делает программы универсальными. Функции, доступные в библиотеки “os”: создание новой папки в текущей, возможность сменить директорию скрипта, удаления файлов, изменение имени файлов или папок, открытие файл в ассоциированной с ним программе, получение кортежа с именем пути к файлу.

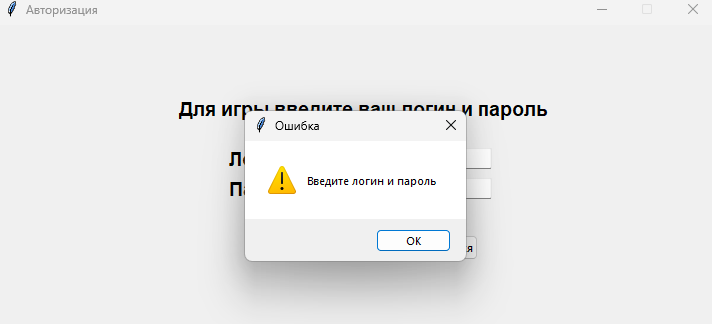
Библиотека “tkinter 3.6” – предоставляет возможность создания графических интерфейсов с помощью оконо, различных виджетов и команд. Основные виджеты библиотеки: Label, Button, Entry, Listbox, Canvas, Listbox.

**4. Сообщения**

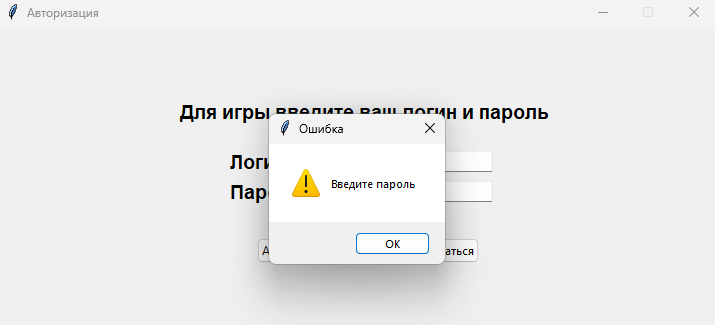
Пользователь не найден или введен неверный пароль



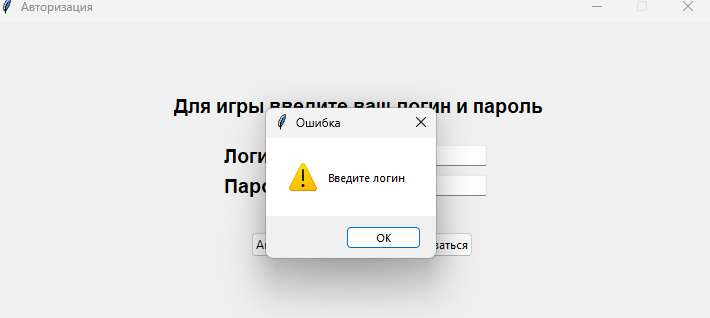
Поле для ввода логина и пароля пустое



Поле для ввода пароля пустое



Поле для ввода логина пустое



### **Текст программы**

import os  
import hashlib  
from tkinter import \*  
from tkinter import ttk  
from tkinter import messagebox  
  
  
def dismiss(win):  
 win.grab\_release()  
 win.destroy()  
  
  
class Checkers:  
 def \_\_init\_\_(self, main):  
 self.deck = None  
 self.first\_click = True  
 self.main = main  
 self.users = {}  
 self.account = {}  
 self.entry\_login = ttk.Entry(width=30, justify='center')  
 self.entry\_password = ttk.Entry(width=30, justify='center')  
 self.txt\_login = Label(text='Логин', font='Arial 14 bold')  
 self.txt\_password = Label(text='Пароль', font='Arial 14 bold')  
 self.button\_registration = ttk.Button(text='Зарегистрироваться', command=lambda: self.regist())  
 self.txt = Label(text='Для игры введите ваш логин и пароль', font='Arial 14 bold')  
 self.button\_autorisation = ttk.Button(text='Авторизоваться', command=lambda: self.authorization())  
  
 self.txt.place(x=180, y=70)  
 self.txt\_login.place(x=230, y=120)  
 self.txt\_password.place(x=230, y=150)  
 self.entry\_login.place(x=310, y=123)  
 self.entry\_password.place(x=310, y=153)  
 self.button\_autorisation.place(x=260, y=210)  
 self.button\_registration.place(x=360, y=210)  
  
 self.x1 = 0  
 self.y1 = 0  
 self.x2 = 0  
 self.y2 = 0  
  
 self.field\_model = [[0, -1, 0, -1, 0, -1, 0, -1, 0, -1],  
 [-1, 0, -1, 0, -1, 0, -1, 0, -1, 0],  
 [ 0, -1, 0, -1, 0, -1, 0, -1, 0, -1],  
 [-1, 0, -1, 0, -1, 0, -1, 0, -1, 0],  
 [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  
 [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  
 [ 0, 1, 0, 1, 0, 1, 0, 1, 0, 1],  
 [ 1, 0, 1, 0, 1, 0, 1, 0, 1, 0],  
 [ 0, 1, 0, 1, 0, 1, 0, 1, 0, 1],  
 [ 1, 0, 1, 0, 1, 0, 1, 0, 1, 0]]  
  
 self.white\_checker = PhotoImage(file="sprites/White.png")  
 self.black\_checker = PhotoImage(file="sprites/Black.png")  
 self.white\_queen = PhotoImage(file="sprites/White\_Queen.png")  
 self.black\_queen = PhotoImage(file="sprites/Black\_Queen.png")  
 self.capture = PhotoImage(file="sprites/capture.png")  
 self.field = PhotoImage(file="sprites/board.png")  
 self.white\_move = True  
 self.black\_move = False  
 self.first\_move = True  
 self.move\_without\_attack = False  
  
 *# Методы для регистрации и авторизации.* def authorization(self):  
 login = self.entry\_login.get()  
 password = self.entry\_password.get()  
 password\_code = hashlib.sha256(password.encode()).hexdigest()  
  
 if len(login) == 0 and len(password) == 0:  
 messagebox.showwarning(title="Ошибка", message="Введите логин и пароль")  
  
 elif len(login) == 0 and len(password) != 0:  
 messagebox.showwarning(title="Ошибка", message="Введите логин")  
  
 elif len(login) != 0 and len(password) == 0:  
 messagebox.showwarning(title="Ошибка", message="Введите пароль")  
  
 else:  
 file = open('save/accounts.txt', 'r+')  
 a = file.readline()[:-1].split(' ')  
  
 while True:  
 if a != ['']:  
 self.users[a[0]] = a[1]  
 a = file.readline()[:-1].split(' ')  
 else:  
 break  
  
 flag\_reg = False  
 flag\_password = True  
 for i in self.users.items():  
 login\_check, password\_check = i  
 if login == login\_check and password\_code == password\_check:  
 flag\_reg = True  
 break  
 elif login == login\_check and password\_code != password:  
 flag\_password = False  
  
 if flag\_reg:  
 for widget in self.main.winfo\_children():  
 widget.destroy()  
  
 win = Toplevel()  
 win.geometry('400x120+760+420')  
 win.title('Успех')  
 win.grab\_set()  
 win.protocol('WM\_DELETE\_WINDOW', lambda: dismiss(win))  
  
 Label(win, text='Вы успешно авторизовались', font='Arial 14 bold').place(x=60, y=20)  
 ttk.Button(win, text='Играть', command=lambda: self.drawing\_field(win)).place(x=160, y=70)  
  
 elif not flag\_password:  
 messagebox.showwarning(title="Ошибка", message="Неверный пароль")  
 else:  
 messagebox.showwarning(title="Ошибка", message="Такого аккаунта не существует")  
  
 def regist(self):  
 def registrate():  
 s\_l = login.get()  
 s\_p = password.get()  
 password\_code = hashlib.sha256(s\_p.encode()).hexdigest()  
  
 if len(s\_l) == 0 or len(s\_p) == 0:  
 messagebox.showwarning(title='Ошибка', message='Поле заполнения пусто')  
 else:  
 file = open('save/accounts.txt', 'r+')  
 a = file.readline()[:-1].split(' ')  
  
 while True:  
 if a != ['']:  
 self.account[a[0]] = a[1]  
 a = file.readline()[:-1].split(' ')  
 else:  
 break  
  
 f\_reg = False  
  
 for i in self.account.items():  
 l, p = i  
 if s\_l == l:  
 f\_reg = True  
  
 if not f\_reg:  
 file = open('save/accounts.txt', 'r+')  
 file.seek(0, os.SEEK\_END)  
 file.write(f'{s\_l} {password\_code}\n')  
 file.close()  
  
 for widget in win.winfo\_children():  
 widget.destroy()  
  
 Label(win, text='Вы успешно зарегистрировались', font='Arial 14 bold').place(x=140, y=120)  
 win.after(800, lambda: (win.destroy(), win.grab\_release()))  
 else:  
 messagebox.showwarning(title='Ошибка', message='Такой аккаунт уже существует')  
  
 win = Toplevel()  
 win.geometry('600x300+660+350')  
 win.title('Регистрация')  
 win.resizable(False, False)  
 win.protocol('WM\_DELETE\_WINDOW', lambda: dismiss(win))  
 win.grab\_set()  
  
 login = ttk.Entry(win, width=30, justify='center')  
 password = ttk.Entry(win, width=30, justify='center')  
 txt\_login = Label(win, text='Логин', font='Arial 14 bold')  
 txt\_password = Label(win, text='Пароль', font='Arial 14 bold')  
 txt = Label(win, text='Введите желаемые логин и пароль', font='Arial 14 bold')  
 button\_reg = ttk.Button(win, text='Зарегистрироваться', command=lambda: registrate())  
  
 txt.place(x=135, y=70)  
 txt\_login.place(x=170, y=120)  
 txt\_password.place(x=170, y=150)  
 login.place(x=250, y=123)  
 password.place(x=250, y=153)  
 button\_reg.place(x=245, y=210)  
  
 *# Метод для вывода доски* def drawing\_field(self, win=None):  
 win.after(200, lambda: (win.destroy(), win.grab\_release()))  
 self.main.title("Фризские шашки")  
 self.main.geometry("870x870+500+50")  
 cell\_sz = 80  
 row = 10  
 col = 10  
  
 self.deck = Canvas(self.main, width=870, height=870)  
 cell\_colors = ["#FFDDBB", "#552B00"]  
 color\_index = 0  
  
 self.deck.create\_image(0, 0, anchor="nw", image=self.field)  
  
 for rows in range(row):  
 for cols in range(col):  
 x1, y1 = cols \* cell\_sz, rows \* cell\_sz  
 cell = self.field\_model[rows][cols]  
 if cell < 0:  
 if cell == -1:  
 self.deck.create\_image(x1 + 37, y1 + 37, anchor="nw", image=self.black\_checker)  
 else:  
 pass  
  
 elif cell > 0:  
 if cell == 1:  
 self.deck.create\_image(x1 + 37, y1 + 37, anchor="nw", image=self.white\_checker)  
 else:  
 pass  
  
 self.deck.pack()  
 self.interact()  
  
 *# Метод для отрисовки ходов* def drawing\_move(self):  
 self.deck.delete("all")  
 cell\_colors = ["#FFDDBB", "#552B00"]  
 color\_index = 0  
 cell\_sz = 80  
 row = 10  
 col = 10  
  
 self.deck.create\_image(0, 0, anchor="nw", image=self.field)  
  
 for rows in range(row):  
 for cols in range(col):  
 x1, y1 = cols \* cell\_sz, rows \* cell\_sz  
 cell = self.field\_model[rows][cols]  
 if cell < 0:  
 if cell == -1:  
 self.deck.create\_image(x1 + 37, y1 + 37, anchor="nw", image=self.black\_checker)  
 elif cell == -2:  
 self.deck.create\_image(x1 + 37, y1 + 37, anchor="nw", image=self.black\_queen)  
  
 elif cell > 0:  
 if cell == 1:  
 self.deck.create\_image(x1 + 37, y1 + 37, anchor="nw", image=self.white\_checker)  
 elif cell == 2:  
 self.deck.create\_image(x1 + 37, y1 + 37, anchor="nw", image=self.white\_queen)  
  
 self.deck.pack()  
 self.checkers\_end()  
  
 *# Метод для взаимодействия с пользователями* def interact(self):  
 self.x1 = 0  
 self.y1 = 0  
 self.x2 = 0  
 self.y2 = 0  
  
 def queen\_check(x2, y2):  
 if self.white\_move:  
 if self.x2 == 0:  
 self.field\_model[x2][y2] = 2  
 else:  
 if self.x2 == 9:  
 self.field\_model[x2][y2] = -2  
  
 def attack\_check(x1, y1, x2, y2):  
 coord\_checkers\_user = [x1, y1, x2, y2]  
 coord\_checkers = []  
 if self.white\_move:  
 flag\_white\_attack = False  
 for x in range(10):  
 for y in range(10):  
 if self.field\_model[x][y] == 1:  
 for i, j in (1, -1), (-1, 1), (-1, -1), (1, 1), (0, 2), (0, -2), (2, 0), (-2, 0):  
 if 0 <= (x + i) <= 9 and 0 <= (y + j) <= 9:  
 if self.field\_model[x + i][y + j] == -1 or self.field\_model[x + i][y + j] == -2:  
 if 0 <= (x + i \* 2) <= 9 and 0 <= (y + j \* 2) <= 9:  
 if self.field\_model[x + i + i][y + j + j] == 0:  
 flag\_white\_attack = True  
 coord\_checkers.append([x, y, x + i + i, y + j + j])  
  
 elif self.field\_model[x][y] == 2:  
 for d in range(4):  
 h = 0  
 if d == 0:  
 i = 1  
 j = 1  
 elif d == 1:  
 i = 1  
 j = -1  
 elif d == 2:  
 i = -1  
 j = 1  
 elif d == 3:  
 i = -1  
 j = -1  
  
 while 0 <= x + (i \* h) + i <= 9 and 0 <= y + (j \* h) + j <= 9:  
 if self.field\_model[x + (i \* h)][y + (j \* h)] == -1 or \  
 self.field\_model[x + (i \* h)][y + (j \* h)] == -2:  
 if self.field\_model[x + (i \* h) + i][y + (j \* h) + j] == 0:  
 flag\_white\_attack = True  
 h += 1  
 while 0 <= x + (i \* h) <= 9 and 0 <= y + (j \* h) <= 9:  
 if self.field\_model[x + (i \* h)][y + (j \* h)] == 0:  
 coord\_checkers.append([x, y, x + (i \* h), y + (j \* h)])  
 else:  
 break  
 h += 1  
 else:  
 break  
 h += 1  
  
 if not flag\_white\_attack:  
 return True  
 else:  
 if coord\_checkers\_user in coord\_checkers:  
 return True  
  
 else:  
 flag\_black\_attack = False  
 for x in range(10):  
 for y in range(10):  
 if self.field\_model[x][y] == -1:  
 for i, j in (1, -1), (-1, 1), (-1, -1), (1, 1), (0, 2), (0, -2), (2, 0), (-2, 0):  
 if 0 <= (x + i) <= 9 and 0 <= (y + j) <= 9:  
 if self.field\_model[x + i][y + j] == 1 or self.field\_model[x + i][y + j] == 2:  
 if 0 <= (x + i \* 2) <= 9 and 0 <= (y + j \* 2) <= 9:  
 if self.field\_model[x + i + i][y + j + j] == 0:  
 flag\_black\_attack = True  
 coord\_checkers.append([x, y, x + i + i, y + j + j])  
 elif self.field\_model[x][y] == -2:  
 for d in range(4):  
 h = 0  
 if d == 0:  
 i = 1  
 j = 1  
 elif d == 1:  
 i = 1  
 j = -1  
 elif d == 2:  
 i = -1  
 j = 1  
 elif d == 3:  
 i = -1  
 j = -1  
  
 while 0 <= x + (i \* h) + i <= 9 and 0 <= y + (j \* h) + j <= 9:  
 if self.field\_model[x + (i \* h)][y + (j \* h)] == 1 or \  
 self.field\_model[x + (i \* h)][y + (j \* h)] == 2:  
 if self.field\_model[x + (i \* h) + i][y + (j \* h) + j] == 0:  
 flag\_black\_attack = True  
 h += 1  
 while 0 <= x + (i \* h) <= 9 and 0 <= y + (j \* h) <= 9:  
 if self.field\_model[x + (i \* h)][y + (j \* h)] == 0:  
 coord\_checkers.append([x, y, x + (i \* h), y + (j \* h)])  
 else:  
 break  
 h += 1  
 else:  
 break  
 h += 1  
  
 if not flag\_black\_attack:  
 return True  
 else:  
 if coord\_checkers\_user in coord\_checkers:  
 return True  
 else:  
 return False  
  
 def multijump\_check(x2, y2):  
 if self.white\_move:  
 if self.field\_model[x2][y2] == 1:  
 for i, j in (1, -1), (-1, 1), (-1, -1), (1, 1), (0, 2), (0, -2), (2, 0), (-2, 0):  
 if 0 < (x2 + i) < 9 and 0 < (y2 + j) < 9:  
 if self.field\_model[x2 + i][y2 + j] == -1 or self.field\_model[x2 + i][y2 + j] == -2:  
 if 0 <= (x2 + i \* 2) <= 9 and 0 <= (y2 + j \* 2) <= 9:  
 if self.field\_model[x2 + i + i][y2 + j + j] == 0:  
 return True  
 elif self.field\_model[x2][y2] == 2:  
 for d in range(4):  
 if d == 0:  
 i = 1  
 j = 1  
 elif d == 1:  
 i = 1  
 j = -1  
 elif d == 2:  
 i = -1  
 j = 1  
 elif d == 3:  
 i = -1  
 j = -1  
  
 for h in range(1, 9):  
 if 0 < x2 + (i \* h) < 9 and 0 < y2 + (j \* h) < 9:  
 if self.field\_model[x2 + (i \* h)][y2 + (j \* h)] == -1 or \  
 self.field\_model[x2 + (i \* h)][y2 + (j \* h)] == -2:  
 if self.field\_model[x2 + (i \* h) + i][y2 + (j \* h) + j] == 0:  
 return True  
 else:  
 break  
 else:  
 break  
  
 else:  
 if self.field\_model[x2][y2] == -1:  
 for i, j in (1, -1), (-1, 1), (-1, -1), (1, 1), (0, 2), (0, -2), (2, 0), (-2, 0):  
 if 0 < (x2 + i) < 9 and 0 < (y2 + j) < 9:  
 if self.field\_model[x2 + i][y2 + j] == 1 or self.field\_model[x2 + i][y2 + j] == 2:  
 if 0 <= (x2 + i \* 2) <= 9 and 0 <= (y2 + j \* 2) <= 9:  
 if self.field\_model[x2 + i + i][y2 + j + j] == 0:  
 return True  
 elif self.field\_model[x2][y2] == -2:  
 for d in range(4):  
 if d == 0:  
 i = 1  
 j = 1  
 elif d == 1:  
 i = 1  
 j = -1  
 elif d == 2:  
 i = -1  
 j = 1  
 elif d == 3:  
 i = -1  
 j = -1  
  
 for h in range(1, 9):  
 if 0 < x2 + (i \* h) < 9 and 0 < y2 + (j \* h) < 9:  
 if self.field\_model[x2 + (i \* h)][y2 + (j \* h)] == 1 or \  
 self.field\_model[x2 + (i \* h)][y2 + (j \* h)] == 2:  
 if self.field\_model[x2 + (i \* h) + i][y2 + (j \* h) + j] == 0:  
 return True  
 else:  
 break  
 else:  
 break  
 return False  
  
 def move\_check(x1, y1, x2, y2):  
 self.move\_without\_attack = False  
 if self.field\_model[x2][y2] != 0:  
 return False  
  
 if self.field\_model[x1][y1] == 2 or self.field\_model[x1][y1] == -2:  
 if abs(x1 - x2) == abs(y1 - y2):  
 count\_queen = 0  
 coord\_checker = []  
 i = 0  
 j = 0  
  
 if x1 < x2:  
 i = 1  
 elif x2 < x1:  
 i = -1  
 if y1 < y2:  
 j = 1  
 elif y2 < y1:  
 j = -1  
  
 for h in range(1, abs(x2 - x1) + 1):  
 if self.field\_model[x1 + (i \* h)][y1 + (j \* h)] == 1 or self.field\_model[x1 + (i \* h)][  
 y1 + (j \* h)] == 2:  
 if self.field\_model[x1][y1] == 2:  
 return False  
 else:  
 count\_queen += 1  
 coord\_checker = [x1 + (i \* h), y1 + (j \* h)]  
  
 if self.field\_model[x1 + (i \* h)][y1 + (j \* h)] == -1 or self.field\_model[x1 + (i \* h)][  
 y1 + (j \* h)] == -2:  
 if self.field\_model[x1][y1] == -2:  
 return False  
 else:  
 count\_queen += 1  
 coord\_checker = [x1 + (i \* h), y1 + (j \* h)]  
 print(count\_queen, "г")  
 if count\_queen == 1:  
 self.field\_model[coord\_checker[0]][coord\_checker[1]] = 0  
 self.field\_model[x2][y2] = self.field\_model[x1][y1]  
 self.field\_model[x1][y1] = 0  
 return True  
  
 elif count\_queen > 1:  
 return False  
  
 else:  
 if attack\_check(self.x1, self.y1, self.x2, self.y2):  
 self.field\_model[x2][y2] = self.field\_model[x1][y1]  
 self.field\_model[x1][y1] = 0  
 self.move\_without\_attack = True  
 return True  
  
 else:  
 if abs(x2 - x1) == 1 and abs(y2 - y1) == 1:  
 if self.black\_move:  
 if x2 < x1 and self.field\_model[x1][y1] == -1:  
 return False  
 elif self.white\_move:  
 if x1 < x2 and self.field\_model[x1][y1] == 1:  
 return False  
 if attack\_check(self.x1, self.y1, self.x2, self.y2):  
 self.field\_model[x2][y2] = self.field\_model[x1][y1]  
 self.field\_model[x1][y1] = 0  
 self.move\_without\_attack = True  
 return True  
 elif abs(x2 - x1) == 2 and abs(y2 - y1) == 2:  
 if self.field\_model[(x1 + x2) // 2][(y1 + y2) // 2] == 0 or self.field\_model[(x1 + x2) // 2][  
 (y1 + y2) // 2] == self.field\_model[x1][y1]:  
 return False  
 self.field\_model[x2][y2] = self.field\_model[x1][y1]  
 self.field\_model[x1][y1] = 0  
 self.field\_model[(x1 + x2) // 2][(y1 + y2) // 2] = 0  
 return True  
 elif abs(x2 - x1) == 4 and abs(y2 - y1) == 0:  
 if self.field\_model[(x1 + x2) // 2][(y1 + y2) // 2] == 0 or self.field\_model[(x1 + x2) // 2][  
 (y1 + y2) // 2] == self.field\_model[x1][y1]:  
 return False  
 self.field\_model[x2][y2] = self.field\_model[x1][y1]  
 self.field\_model[x1][y1] = 0  
 self.field\_model[(x1 + x2) // 2][(y1 + y2) // 2] = 0  
 return True  
 elif abs(x2 - x1) == 0 and abs(y2 - y1) == 4:  
 if self.field\_model[(x1 + x2) // 2][(y1 + y2) // 2] == 0 or self.field\_model[(x1 + x2) // 2][  
 (y1 + y2) // 2] == self.field\_model[x1][y1]:  
 return False  
 self.field\_model[x2][y2] = self.field\_model[x1][y1]  
 self.field\_model[x1][y1] = 0  
 self.field\_model[(x1 + x2) // 2][(y1 + y2) // 2] = 0  
 return True  
 return False  
  
 def click\_event\_capt(event):  
 if 35 < event.x < 835 and 0 < event.y < 835:  
 self.y2 = (event.x - 35) // 80  
 self.x2 = (event.y - 35) // 80  
 if self.first\_move:  
 if self.x1 == self.x2 and self.y1 == self.y2:  
 self.main.bind("<Button-1>", click\_event)  
 self.drawing\_move()  
  
 else:  
 if move\_check(self.x1, self.y1, self.x2, self.y2):  
 print((self.x1, self.y1, self.x2, self.y2))  
 queen\_check(self.x2, self.y2)  
 self.drawing\_move()  
 if not self.move\_without\_attack:  
 if multijump\_check(self.x2, self.y2):  
 self.first\_move = False  
 self.x1, self.y1 = self.x2, self.y2  
 self.deck.create\_image(self.y1 \* 80 + 35, self.x1 \* 80 + 35, anchor="nw",  
 image=self.capture)  
 self.main.bind("<Button-1>", click\_event\_capt)  
 else:  
 self.white\_move, self.black\_move = not self.white\_move, not self.black\_move  
 self.interact()  
 else:  
 self.white\_move, self.black\_move = not self.white\_move, not self.black\_move  
 self.interact()  
  
 elif not self.first\_move:  
 if move\_check(self.x1, self.y1, self.x2, self.y2) and not self.move\_without\_attack:  
 queen\_check(self.x2, self.y2)  
 self.drawing\_move()  
 if multijump\_check(self.x2, self.y2):  
 self.x1, self.y1 = self.x2, self.y2  
 self.deck.create\_image(self.y1 \* 80 + 35, self.x1 \* 80 + 35, anchor="nw",  
 image=self.capture)  
 self.main.bind("<Button-1>", click\_event\_capt)  
 else:  
 self.drawing\_move()  
 self.first\_move = True  
 self.white\_move, self.black\_move = not self.white\_move, not self.black\_move  
 self.interact()  
  
 def click\_event(event):  
 if 35 < event.x < 835 and 35 < event.y < 835:  
  
 self.y1 = (event.x - 35) // 80  
 self.x1 = (event.y - 35) // 80  
  
 if self.white\_move and (  
 self.field\_model[self.x1][self.y1] == 1 or self.field\_model[self.x1][self.y1] == 2):  
 self.deck.create\_image(self.y1 \* 80 + 35, self.x1 \* 80 + 35, anchor="nw", image=self.capture)  
 self.main.bind("<Button-1>", click\_event\_capt)  
  
 elif self.black\_move and (  
 self.field\_model[self.x1][self.y1] == -1 or self.field\_model[self.x1][self.y1] == -2):  
 self.deck.create\_image(self.y1 \* 80 + 35, self.x1 \* 80 + 35, anchor="nw", image=self.capture)  
 self.main.bind("<Button-1>", click\_event\_capt)  
  
 self.main.bind("<Button-1>", click\_event)  
  
 *# Метод для проверки окончания игры* def checkers\_end(self):  
 flag\_white\_move = False  
 flag\_black\_move = False  
 for x in range(10):  
 for y in range(10):  
 if self.field\_model[x][y] == 1 or self.field\_model[x][y] == 2:  
 for i, j in (1, -1), (-1, 1), (-1, -1), (1, 1):  
 if 0 <= (x + i) <= 7 and 0 <= (y + j) <= 9:  
  
 if self.field\_model[x + i][y + j] == -1 or self.field\_model[x + i][y + j] == -2:  
 if 0 <= (x + i \* 2) <= 7 and 0 < (y + j \* 2) <= 9:  
 if self.field\_model[x + i + i][y + j + j] == 0:  
 flag\_white\_move = True  
 break  
 elif self.field\_model[x + i][y + j] == 0:  
 flag\_white\_move = True  
 break  
  
 elif self.field\_model[x][y] == -1 or self.field\_model[x][y] == -2:  
 for i, j in (1, -1), (-1, 1), (-1, -1), (1, 1):  
 if 0 <= (x + i) <= 7 and 0 <= (y + j) <= 9:  
 if self.field\_model[x + i][y + j] == 1 or self.field\_model[x + i][y + j] == 2:  
 if 0 <= (x + i \* 2) <= 7 and 0 <= (y + j \* 2) <= 9:  
 if self.field\_model[x + i + i][y + j + j] == 0:  
 flag\_black\_move = True  
 break  
 elif self.field\_model[x + i][y + j] == 0:  
 flag\_black\_move = True  
 break  
  
 if not flag\_white\_move:  
 self.end\_game("Чёрных")  
 elif not flag\_black\_move:  
 self.end\_game("Белых")  
 elif not flag\_white\_move and not flag\_black\_move:  
 self.end\_game("Ничья")  
  
 *# Метод для вывода результатов окончания игры* def end\_game(self, winner):  
 winner\_window = Tk()  
 winner\_window.geometry("150x150+600+300")  
 winner\_window.protocol('WM\_DELETE\_WINDOW', lambda: dismiss(winner\_window))  
 winner\_window.grab\_set()  
  
 self.white\_move = False  
 self.black\_move = True  
 self.first\_move = True  
  
 self.field\_model = [[0, -1, 0, -1, 0, -1, 0, -1, 0, -1],  
 [-1, 0, -1, 0, -1, 0, -1, 0, -1, 0],  
 [0, -1, 0, -1, 0, -1, 0, -1, 0, -1],  
 [-1, 0, -1, 0, -1, 0, -1, 0, -1, 0],  
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  
 [0, 1, 0, 1, 0, 1, 0, 1, 0, 1],  
 [1, 0, 1, 0, 1, 0, 1, 0, 1, 0],  
 [0, 1, 0, 1, 0, 1, 0, 1, 0, 1],  
 [1, 0, 1, 0, 1, 0, 1, 0, 1, 0]]  
  
 if winner == "Ничья":  
 Label(winner\_window, text="Ничья").pack()  
 else:  
 Label(winner\_window, text=f"Победа {winner}").pack()  
 ttk.Button(winner\_window, text="Заново",  
 command=lambda: (winner\_window.grab\_release(), winner\_window.destroy(), self.drawing\_move())).pack()  
  
  
root = Tk()  
root.title("Авторизация")  
root.geometry('720x300+600+350')  
root.resizable(False, False)  
  
Checkers(root)  
  
root.mainloop()