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»**

**«Компьютерная логическая игра “Рэндзю”»**

**Инв. № подл.**

**Подп. и дата**

**Взам. инв. №**

**Инв. № дубл.**

**Подп. и дата**

Р.02069337.21/839-09 ПЗ-01

Листов 39

**Руководитель разработки**:

доцент каф. ИВК, к.т.н., доцент

*Шишкин Вадим Викторинович*

« » 2022 г.

**Исполнитель**:

студент гр. ИСТбд-22

*Колесников Павел Александрович*

« » 2022 г.

**2022**

Содержание

[Аннотация 2](#_Toc155640781)

[Техническое задание 4](#_Toc155640782)

[Пояснительная записка 9](#_Toc155640783)

[Руководство программиста 16](#_Toc155640784)

[Текст программы 28](#_Toc155640785)

# **Аннотация**

Данный документ представляет собой пояснительную записку на курсовую работу на тему «Компьютерная логическая игра «Рэндзю»». Документ содержит следующие разделы: техническое задание, пояснительная записка и руководство программиста, код программы; в нем излагается постановка задачи и описание реализуемой программы, ее назначение. Документ может быть использован в качестве инструкции для применения рассматриваемого программного средства.

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 «Рэндзю»

**Инв. № подл.**

**Подп. и дата**

**Взам. инв. №**

**Инв. № дубл.**

**Подп. и дата**

Р.02069337.21/839-09 ТЗ-01

Листов 5

**Исполнитель**:

студент гр. ИСТбд-22

*Колесников Павел Александрович*

« » 2022 г.

**2022**

**Введение**

Наименование игры: Рэндзю

Наименование разрабатываемой игры: RENJU

Рэндзю - японская настольная логическая игра для двух игроков. Игроки поочередно размещают свои камни на игровом поле. Первый игрок использует камни черного цвета, а второй игрок - белого. Цель – построить непрерывную горизонтальную, вертикальную или диагональную линию из пяти своих камней быстрее противника. Игра продолжается до достижения одним из игроков пяти рядом стоящих камней или до заполнения всего игрового поля.

Правила игры:

1. Первый ход (ход черных) выполняется строго в центр игрового поля.
2. Второй ход (ход белых) выполняется в радиусе одной клетки от центра игрового поля.
3. Третий ход (ход черных) выполняется в радиусе двух клеток от центра игрового поля.

Приложение предоставляет список возможностей:

1. Регистрацию/Вход
2. Выбор новой или последней сохраненной игры
3. Выбор темы игрового поля, цвета камней, возможности отмены последнего хода.
4. Сохранение и выход из незавершенной игры.

**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 или программному изделию**

**2.1. Функциональное назначение**

**Функциональное назначение**

Требуется разработать однопользовательское десктопное приложение по игре в Рэндзю с графическим интерфейсом в среде Windows.

**2.2 Требования к функциональным характеристикам**

2.2.1 Требования к структуре приложения

Приложение должно быть разработано в виде одного модуля с дополнительными информационными файлами при необходимости.

Приложение должно соответствовать следующим правилам игры.

**Поле и игроки.** Игра ведется между двумя соперниками (пользователь-компьютер) камнями разного цвета (чёрные и белые) на квадратном поле, представляющее собой пересечение 15 вертикальных и 15 горизонтальных линий. На образованные линиями пересечения игроками ставятся камни. Из 225 пересечений игрового поля помечены жирным: одна центральная (для первого хода в партии) и линии составляющие 2 радиуса вокруг центральной точки.

**Специальные термины.** В дальнейшем будут использоваться такие термины:

Двойка - линия из двух камней, расположенные последовательно по горизонтали, вертикали или диагонали.

Тройка - линия из трех камней, расположенные последовательно по горизонтали, вертикали или диагонали.

Четвёртка - линия из четырех камней, расположенные последовательно по горизонтали, вертикали или диагонали.

Пятёрка - линия из пяти камней, расположенные последовательно по горизонтали, вертикали или диагонали, приводящая к победе.

**Порядок ходов.** Игроки совершают ходы поочередно. Первый ход делает игрок с чёрными камнями автоматически в центральную точку. После каждым ходом игрок выставляет на доску в любое свободное пересечение линий доски один камень своего цвета.

**Цель игры.** Построить как можно скорее непрерывный ряд из пяти камней своего цвета — по горизонтали, по вертикали или в диагональном направлении. Игра завершается, когда один из игроков составит ряд из пяти камней своего цвета. Игрок, первый составивший ряд из пяти камней своего цвета, считается победителем, второй игрок – проигравшим. Ничья объявляется, когда для обоих соперников отсутствует возможность построения пятёрки и все точки на поле заняты.

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 приемы, проверка правильности и отрисовка хода пользователя;

- проверка окончания игры;

- вычисление, проверка правильности и отрисовка хода компьютера;

- информирование пользователя об окончании игры и победителе.

2.2.3 Требования к организации информационного обеспечения, входных и выходных данных

В приложении должен быть реализован графический интерфейс взаимодействия с пользователем. Изображения камней могут храниться в отдельных графических файлах. Логин и пароль пользователя должны вводиться с клавиатуры. Логины и пароли зарегистрированных пользователей должны храниться в отдельном файле. Пояснительные информационные сообщения для пользователя должны выводиться сбоку игрового поля по ходу игры, либо во всплывающих сообщениях.

**2.3 Требования к надёжности**

Приложение должно быть стабильным и работоспособным, не вызывать сбоев или ошибок. В случае сбоя или ошибки, приложение должно быть способно восстановиться без потери данных.

**2.4 Требования к информационной и программной совместимости**

Операционная система: Windows 10 Домашняя.

Версия языка программирования: Python 3.11

Среда разработки: PyCharm Community Edition 2023.3.4

При создании программы используются встроенные библиотеки “os”, “math”, “sys” и сторонние библиотеки “pygame”, “tkinter”, “random”.

**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

Проект будет храниться в репозитории на сайте github.com по ссылке https://github.com/kepcher2131/KR\_RENJU.git

2.6 3 Сроки хранения

Срок хранения – до июля 2025 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»**

# **ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**Тема:** Компьютерная логическая игра «Рэндзю»

**Инв. № подл.**

**Подп. и дата**

**Взам. инв. №**

**Инв. № дубл.**

**Подп. и дата**

Р.02069337. 21/839-09 ПЗ-01

Листов 7

**Исполнитель**:

студент гр. ИСТбд-22

*Колесников Павел Александрович*

« » 2022 г.

**2022**

**Введение**

Приложение «RENJU» представляет собой игру Рэндзю на 225 пересечениях (15линий в ширину и 15 в длину) для двоих игроков.

Выбранная структура для эмуляции игровой доски в рэндзю, это одномерный список, который представляет собой набор значений координат игрового поля. Первое значение списка отвечает за координату x, а второе за координату y.

Для хранения данных пользователей, выбран список словарей.

Для сохранения массива словарей используется файл с расширение json, так как python словарь и json схожи по формату. Файл перед сохранением шифруется.

**1. Проектная часть**

**1.1 Постановка задачи на разработку приложения**

Определяется заданием на курсовую работу. Детализируется в разработанном техническом задании.

**1.2 Математические методы**

Математические методы не используются.

**1.3 Архитектура и алгоритмы**

1.3.1. Архитектура

Основной модуль

Хранение данных/ запись новых

Авторизация/

регистрация

Отрисовка меню

Отрисовка доски

Отрисовка элементов

Отрисовка ходов

Выбор режима игры

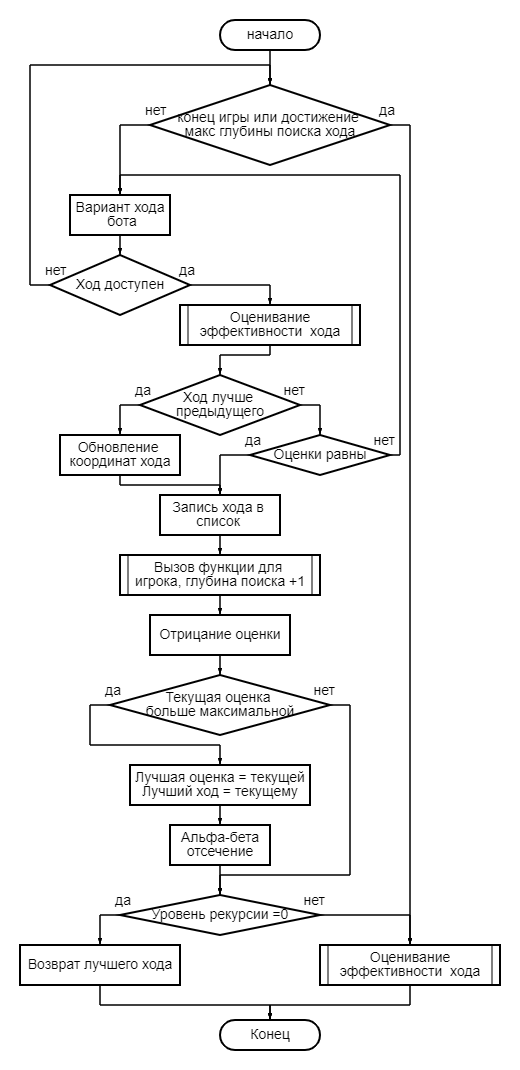
Проверка хода

Обработчик событий

Выбор хода ботом

Проверка конца игры

1.3.2. Алгоритм хода бота основанный на алгоритме Минимакс.



Функция сначала проверяет, не закончена ли игра или достигнута ли максимальная глубина поиска. Если это так, то возвращает значение оценки текущего состояния игры. Затем функция перебирает все возможные ходы для игрока и рекурсивно вызывает себя с измененным состоянием игры. В каждой итерации функция обновляет лучший ход и его оценку, и использует альфа-бета отсечение для улучшения производительности алгоритма. В конце, если текущий уровень рекурсии - верхний уровень, то функция добавляет лучший ход к состоянию и возвращает его. В противном случае возвращает лучшую оценку текущего состояния игры.

1.3.2. Алгоритм шифрования/дешифрования RSA

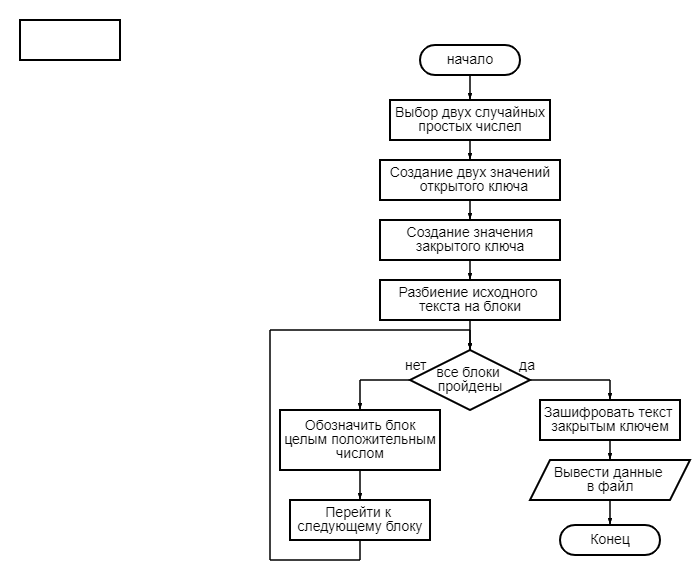


Рисунок 1- Шифрование данных.

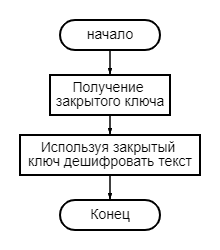
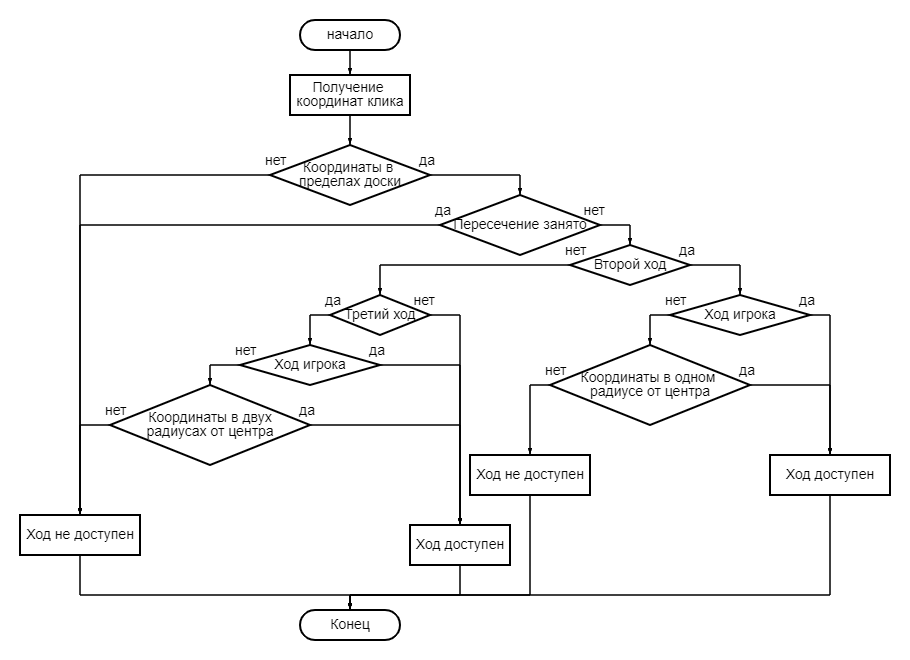
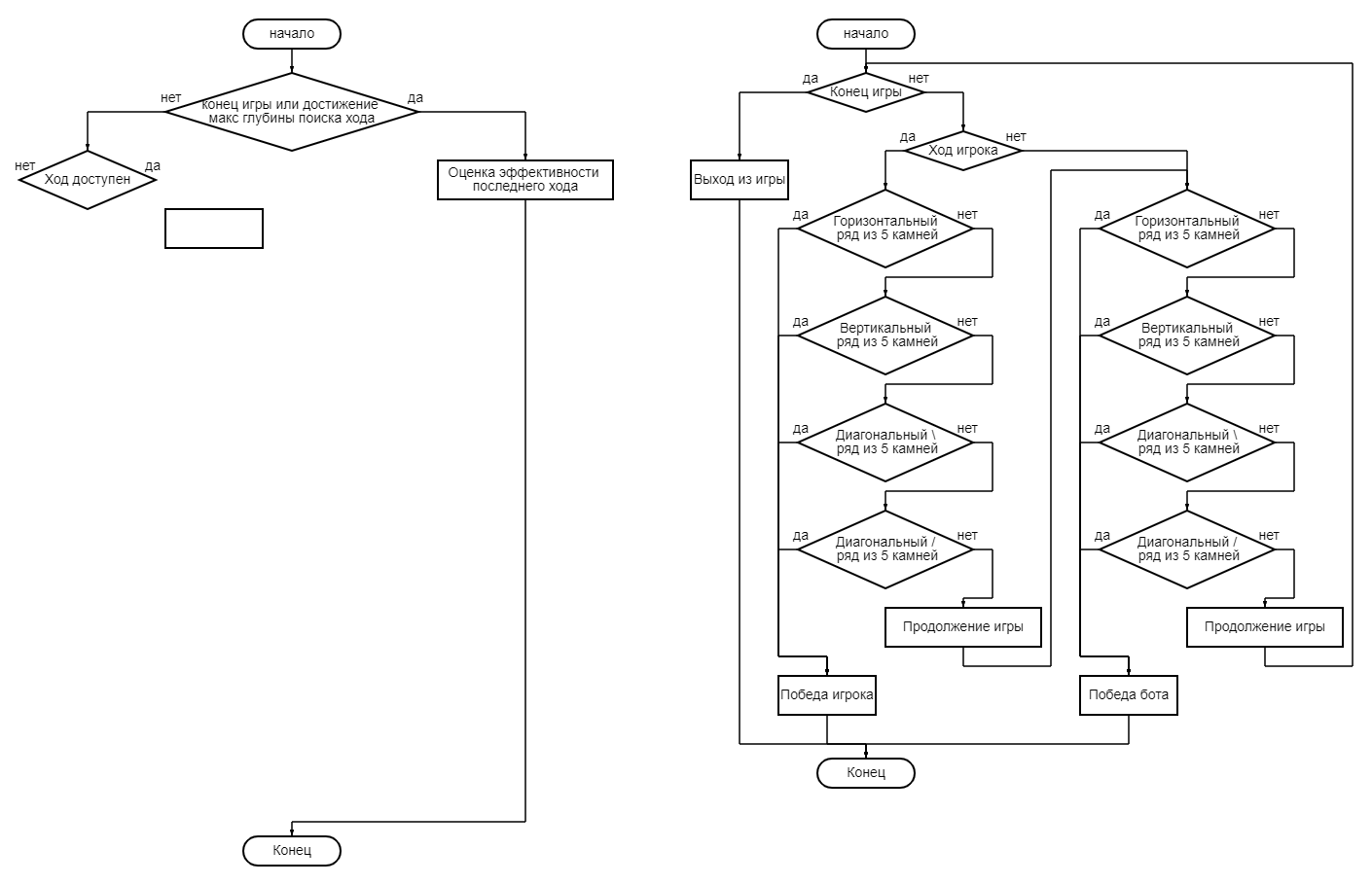


Рисунок 2- Дешифрование данных.

1.3.2. Алгоритм проверки доступности хода

Данный алгоритм регистрации хода необходим для контроля за соблюдением правил игры в рэндзю. Алгоритм принимает на вход координаты выбранной клетки на игровом поле. После получения этих координат, алгоритм проверяет возможность совершения хода в указанную клетку, основываясь на правилах рэндзю. Если ход возможен, игровая шашка размещается в указанной клетке, иначе игрок должен выбрать другое место для своего хода. 

1.3.2. Алгоритм проверки конца игры 

Данный алгоритм необходим для определения завершения игры в Рэндзю. Алгоритм проверяет наличие выигрышной комбинации камней для игрока и бота. В случае выявления выигрышной комбинации, алгоритм запускает процесс завершения игры.

**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 спец 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**

1. Шишкин, В.В. Разработка логических компьютерных игр с графическим интерфейсом в среде Python [Электронный ресурс] / В.В. Шишкин, Д.С. Афонин. – Ульяновск: УлГТУ, 2023. – 89 с. – Режим доступа: для всех пользователей. – URL: <http://lib.ulstu.ru/venec/disk/2023/112.pdf> (дата обращения: 13.3.24).

2. Руководство по программированию на Tkinter и Python [Электронный ресурс]. – Режим доступа: для всех пользователей. – URL: <https://metanit.com/python/tkinter/?ysclid=lq9y8691w1850842283> (дата обращения: 14.3.24)

3. Жданов, Н. А. Алгоритм "минимакс" и его реализация на примере игры "крестики-нолики" / Н. А. Жданов, Ю. М. Бурханова, Ю. О. Воронецкий // Политехнический молодежный журнал. – 2019. – № 5(34). – С. 4. (дата обращения: 13.3.24)

4. Алгоритмы для оценки ходов в играх антагонистического типа / Е. С. Белашова, Р. Р. Хабибулин, Б. Р. Шарипов, К. Ю. Канбекова // Вестник Технологического университета. – 2019. – Т. 22. – № 8. – С. 130-134. (дата обращения: 14.3.24)

5. [RSA (Rivest, Shamir и Adleman) [Электронный ресурс]. – Режим доступа: для всех пользователей. – URL: https://ru.wikipedia.org/wiki/RSA](RSA%20(Rivest,%20Shamir%20и%20Adleman)%20%5bЭлектронный%20ресурс%5d.%20–%20Режим%20доступа:%20для%20всех%20пользователей.%20–%20URL:%20https://ru.wikipedia.org/wiki/RSA) (дата обращения: 14.3.24)

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»**

# **РУКОВОДСТВО ПРОГРАММИСТА**

**Тема:** Компьютерная логическая игра «Рэндзю»

**Инв. № подл.**

**Подп. и дата**

**Взам. инв. №**

**Инв. № дубл.**

**Подп. и дата**

Р.02069337.21/839-09 РП-01

Листов 24

**Исполнитель**:

студент гр. ИСТбд-22

*Колесников Павел Александрович*

« » 2022 г.

**2022**

**1. Назначение и условия применения программы**

**1.1 Назначение и функции, выполняемые приложением**

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

**Поле и игроки.** Игра ведется между двумя соперниками (пользователь-компьютер) камнями разного цвета на квадратной доске размером 15х15 линий.

**Порядок ходов.** Игроки ходят по очереди. Первый ход делает игрок за чёрные камни (пользователь или компьютер; в начале игры даётся выбор) в центр доски. Все дальнейшие ходы игроки могут ставить свои камни на любые пустые клетки.

**Цель игры.** Цель игры - построить непрерывную линию из пяти своих камней по горизонтали, вертикали или диагонали. Игра заканчивается, когда один из игроков достигает этой цели. Игрок, построивший линию из пяти своих камней, считается победителем, а второй - проигравшим.

В приложении предоставляется возможность зарегистрироваться чтобы играть со своим аккаунтом, выбрать тему игрового поля, выбрать возможность отмены последнего хода, сохранить игру, сыграть в логическую игру Рэндзю, в приложении реализованы все правила этой игры: порядок хода игроков, проверка конца игры и вывод победителя.

**1.2 Условия, необходимые для использования приложения**

Рекомендуется к использованию на Windows 10.

При создании программы использовались встроенная библиотека “os”, “math”, “sys” и сторонние библиотеки “pygame”, “tkinter”, “random”.

Разработка велась в PyCharm Community Edition 2023.3.4 на версии языка программирования Python 3.11.

**2. Характеристики программы**

**2.1 Характеристики приложения**

Значимых файлов 3

Значимых строк кода 185/460/540

Структура данных одна – массив.

Алгоритмов 2: minmax, RSA.

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

“random” - используется для генерации случайных чисел и других случайных значений. Она предоставляет функции для генерации случайных целых чисел, чисел с плавающей точкой.

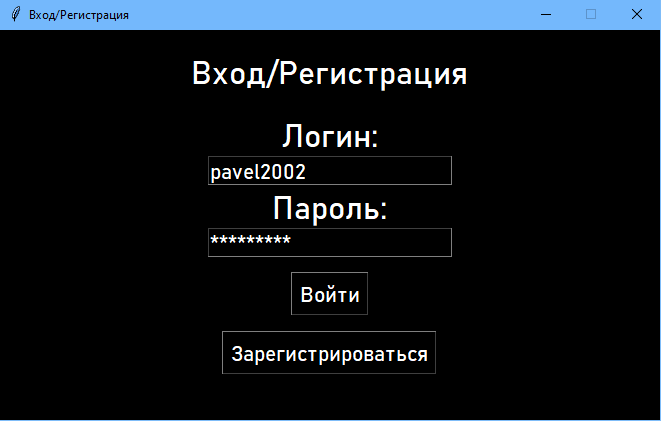
“tkinter” - библиотека для разработки графического интерфейса на языке Python. Методы, включенные в неё, позволяют создавать окна, размещать на них виджеты, настраивать параметры окна и виджетов.

“Pygame”— это библиотека для создания игр на языке программирования Python. В библиотеке собран набор программных средств, помогающих разрабатывать двумерные игры.

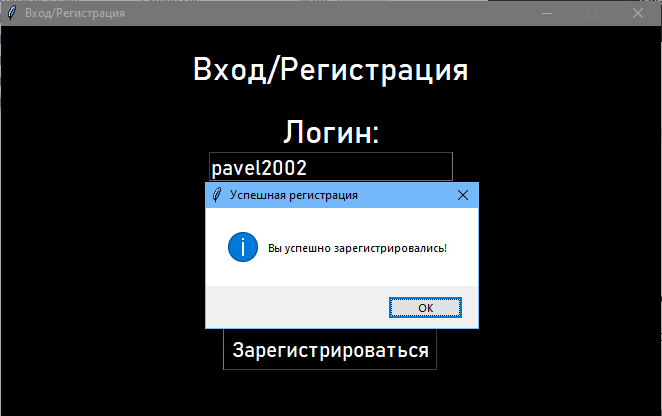
“sys” - модуль в Python предоставляет простые функции, которые позволяют нам напрямую взаимодействовать с интерпретатором

“math” - это встроенная библиотека, которая предоставляет функции и константы для выполнения математических операций. Она позволяет выполнять операции, такие как вычисление корня, возведение в степень, тригонометрические функции и многое другое.

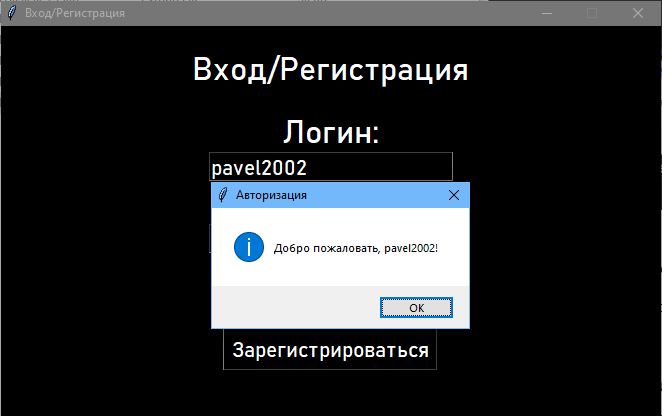
Работа приложения начинается с окна авторизации и регистрации:



Успешная регистрация:



Успешная авторизация:



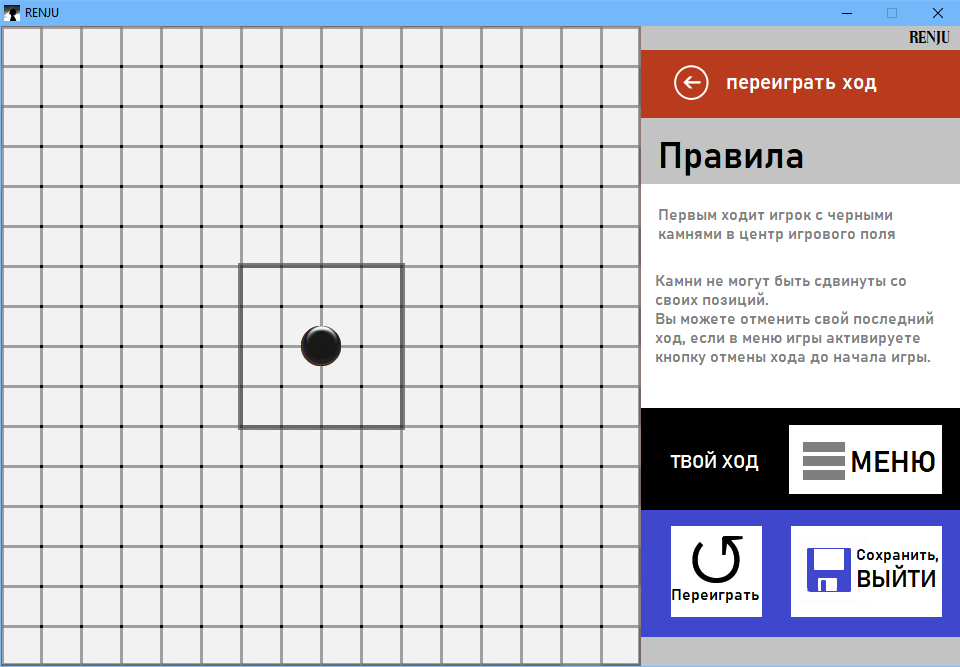
Выбор новой или последней игры:



Выбор темы, возможности переиграть последний ход и цвета камня:



Отрисовка игрового поля и бокового меню с возможностью отмены хода:

Окончание игры и возможность перезапуска:

**2.2 Особенности реализации приложения**

В программе используется структура данных массив, эта структура была выбрана для навигации, отрисовки и проверок ходов. На игровом поле игроки могут размещать свои камни, а массив позволяет эффективно отслеживать, занята ли уже определенная ячейка или она свободна. Кроме того, эта структура данных удобна для проверок победы в различных направлениях на игровом поле.

В игре реализованы следующие возможности: выбор темы игрового поля, возможность переиграть ход, сохранение игры.

**3. Обращение к программе**

1. def show\_game\_window():

Метод, вызываемый после успешной авторизации или регистрации, закрывает текущее окно и открывает окно игры.

1. def login():

Получает введенные пользователем логин и пароль, проверяет их наличие и соответствие данным в файле, выводит сообщения об успешной авторизации или об ошибках.

1. add\_user():

Получает введенные пользователем логин и пароль, проверяет их корректность (длина, наличие цифр и букв), добавляет нового пользователя в файл и выводит сообщение об успешной регистрации.

1. def RSA():

Метод, вызываемый после успешной авторизации или регистрации, выполняет RSA шифрование данных из файла с записанными логин и пароль.

1. def is\_prime(num):

Функция для проверки простоты числа.

1. show\_password(self):

Функция для генерации простых чисел p и q.

1. def calculate\_phi(p, q):

Функция для вычисления функции Эйлера.

1. def mod\_inverse(a, m):

Функция для вычисления обратного числа по модулю.

1. def generate\_keys():

Функция для генерации открытого и закрытого ключей.

1. def encrypt(data, public\_key):

Функция для шифрования данных.

1. def decrypt(encrypted\_data, private\_key):

Функция для дешифрования данных.

1. def mode1(u, enabled, s):

Функция окна игрового поля, в которой реализуется игровой цикл, взаимодействия пользователя с интерфейсом и др нижеперечисленные функции. Первый аргумент в mode1() для определения цвета камня для игрока, 0 и 1 соответственно белый и черный, второй аргумент в mode1() для определения возможности отмены хода, 0 и 1 соответственно True и False, третий аргумент в mode1() для определения сохраненной или новой игры, 1 и 2 соответственно новая и сохр игра.

1. def select():

Инициирует окно выбора параметров игры.

1. def play\_init():

Инициирует первое окно выбора игры после того, как успешно прошла авторизация или регистрация.

1. def attack(player, otherplayer, depth, maxdepth, A, B):

Функция реализует минимаксный алгоритм для игры для двух игроков. Функция «атака» рекурсивно исследует игровое дерево, чтобы найти лучший ход для текущего игрока на заданной глубине, рассматривая возможные ходы и оценивая их потенциальные результаты.

1. def isgameover(play):

Проверяет наличие выигрышной комбинации на игровом поле.

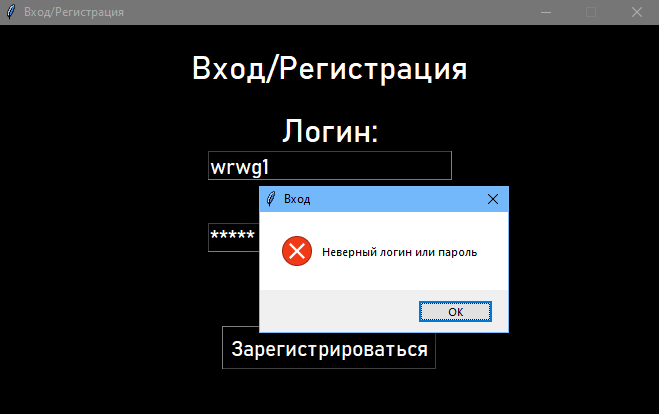
1. def getmoves(pl1, pl2):

Внутри функции определяются возможные ходы для каждого игрока и затем оценивается эффективность каждого из них с помощью функции evaluate(). Результаты оценки сохраняются в переменной s1, а координаты возможных ходов для каждого игрока сохраняются в массиве des.

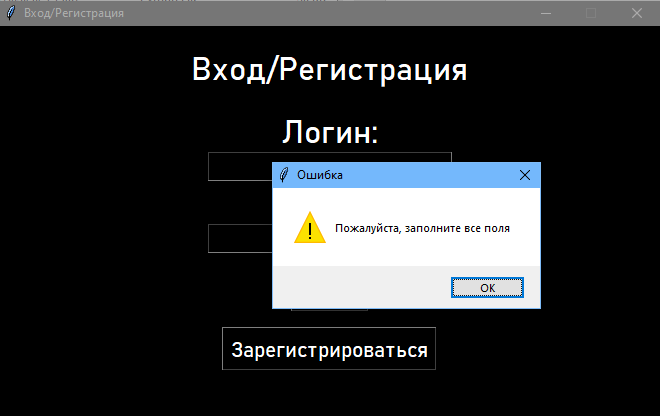
1. def evaluate(pl1, play):

Внутри функции определяются возможные комбинации камней в различных видах ряда и оценивается эффективность каждого из них.

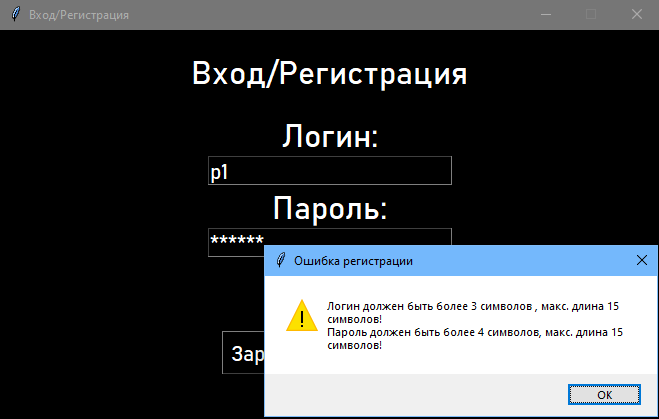
**4. Сообщения**

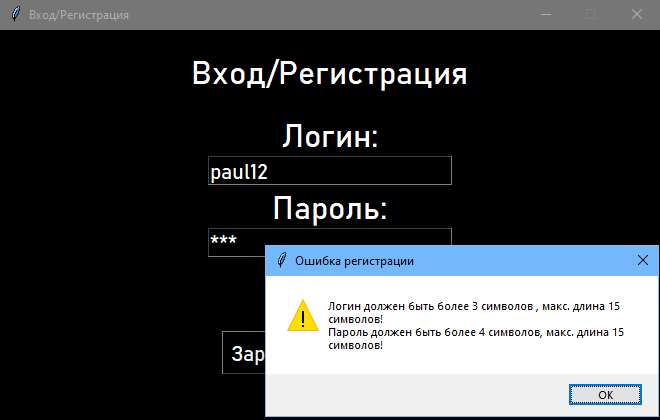
Пользователь не найден или введен неверный пароль:

Поля для ввода пустые:

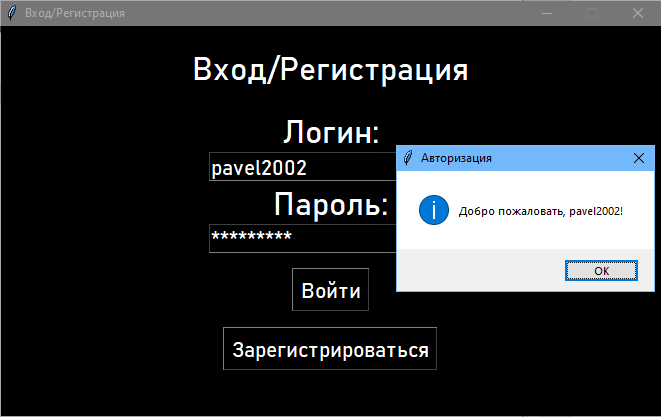


Недопустимый логин пользователя:

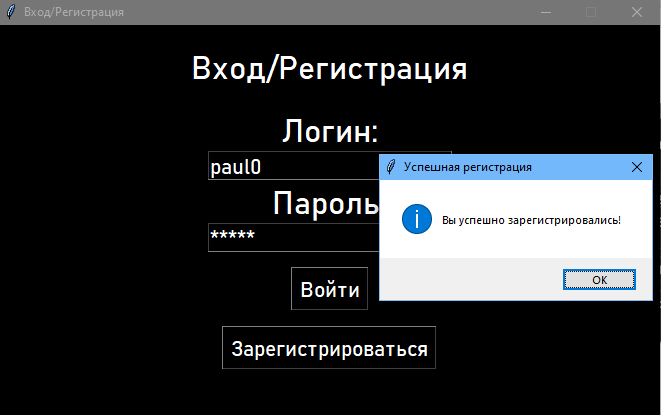


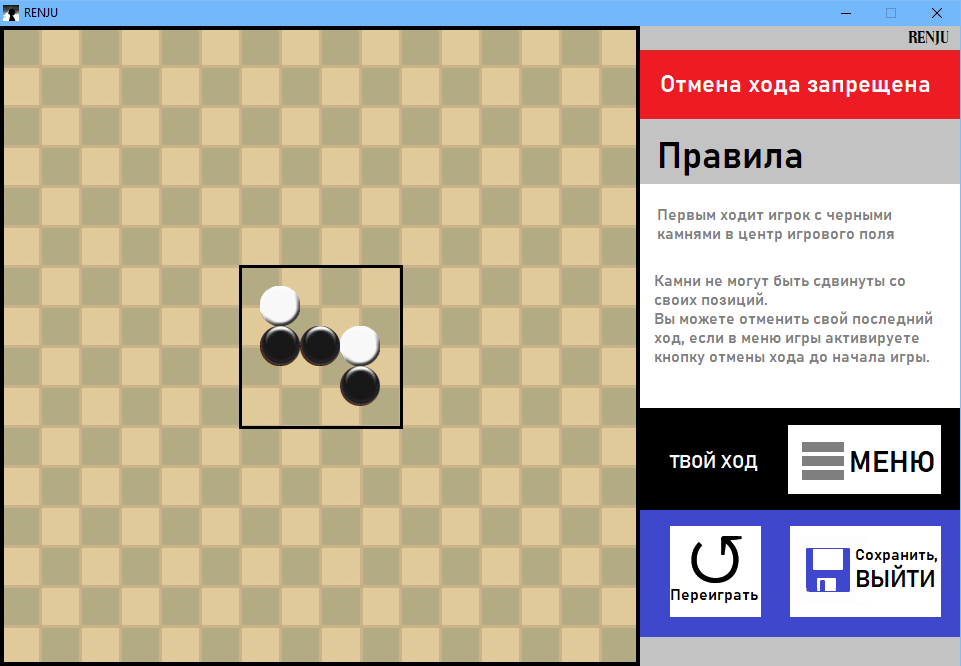


Успешная авторизация:

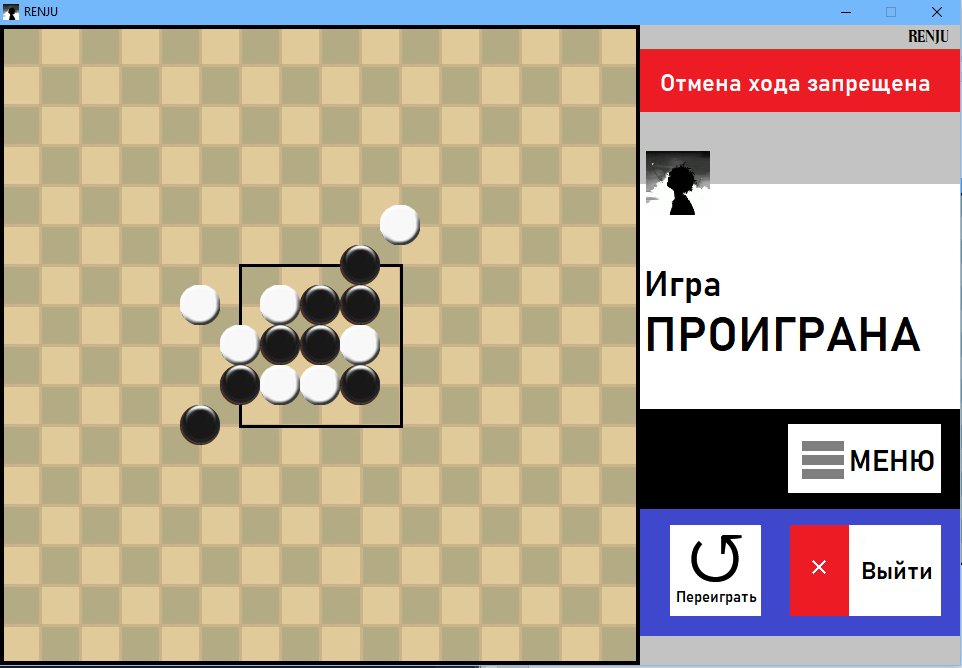


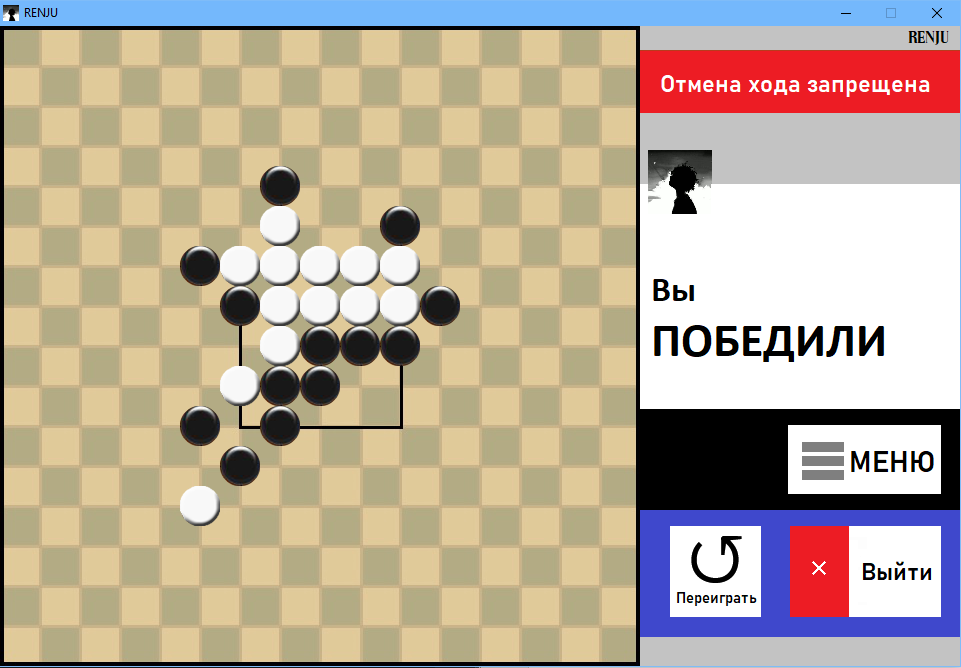
Успешная регистрация:



Вывод сообщения о ходе игрока:

Конец игры:





# **Текст программы**

import algo  
import pygame, sys  
from pygame.locals import \*  
  
# загрузка изображений иконки окна и выбора новой или последней игры  
i\_icon = "images\icon.png"  
mode = "images\selectmode.jpg"  
  
# функция окна игрового поля mode1()  
# первый аргумент в mode1() для определения цвета камня для игрока, 0 и 1 соответственно белый и черный  
# второй аргумент в mode1() для определения возможности отмены хода, 0 и 1 соответственно True и False  
# третий аргумент в mode1() для определения сохраненной или новой игры, 1 и 2 соответственно новая и сохр игра  
def mode1(u, enabled, s):  
 undo = 1  
 # чтение из файла тему игрового поля  
 fread = open('info\\theme.txt', 'r')  
 default = "images\\" + fread.readline()  
 fread.close()  
 backimg = default  
 # цвет камня для игрока, 0 и 1 соответственно белый и черный  
 if u == 0:  
 USER = "images\\blanc.png"  
 AI = "images\\noir.png"  
 elif u == 1:  
 USER = "images\\noir.png"  
 AI = "images\\blanc.png"  
 # загрузка изображений подокон вне игрового поля  
 i\_icon = "images\icon.png"  
 EXIT = "images\exit.png"  
 PANE = "images\pane.jpg"  
 ABOUT = "images\\about.png"  
 WIN = "images\win.png"  
 LOSE = "images\lost.png"  
 YOU = "images\you.png"  
 ME = "images\me.png"  
 UNDO = "images\disabled.png"  
 AGAIN = "images\\again.png"  
 NOTHING = "images\waste.png"  
 # создание массивов для ходов игрока и бота  
 HUMAN = []  
 BOT = []  
 pygame.init()  
 # загрузка изображения иконки окна игры  
 icon = pygame.image.load(i\_icon)  
 pygame.display.set\_icon(icon)  
 # установка экрана на разрешение 960 x 640  
 screen = pygame.display.set\_mode((960, 640), 0, 32)  
 # загрузка изображений темы игрового поля, бокового меню и информации  
 background = pygame.image.load(backimg).convert()  
 pane = pygame.image.load(PANE).convert()  
 about = pygame.image.load(ABOUT).convert\_alpha()  
 win = pygame.image.load(WIN).convert\_alpha()  
 lost = pygame.image.load(LOSE).convert\_alpha()  
 no\_undo = pygame.image.load(UNDO).convert\_alpha()  
 user = pygame.image.load(USER).convert\_alpha()  
 playAgain = pygame.image.load(AGAIN).convert\_alpha()  
 ai = pygame.image.load(AI).convert\_alpha()  
 you = pygame.image.load(YOU).convert\_alpha()  
 me = pygame.image.load(ME).convert\_alpha()  
 nothing = pygame.image.load(NOTHING).convert\_alpha()  
 escape = pygame.image.load(EXIT).convert\_alpha()  
 pygame.display.set\_caption('RENJU')  
 # загрузка экрана игрового поля и боковой панели меню на определенные позиции  
 screen.blit(background, (0, 0))  
 screen.blit(pane, (640, 0))  
 # загрузка кнопки "отмена хода запрещена", если выбран режим без отмены хода  
 if not enabled:  
 screen.blit(no\_undo, (640, 25))  
 # первый ход для новой игры  
 if s == 1:  
 if u == 0:  
 count = 1  
 BOT.append((300, 300))  
 screen.blit(ai, (300, 300))  
 screen.blit(you, (650, 400))  
 elif u == 1:  
 HUMAN.append((300, 300))  
 screen.blit(user, (300, 300))  
 screen.blit(me, (650, 400))  
 count = 1  
 pygame.display.update()  
 # восстановление доски, если игра сохранена  
 elif s == 2:  
 screen.blit(you, (650, 400))  
 fread = open('info\savedGame.txt', 'r')  
 fread.readline()  
 fread.readline()  
 count = int(fread.readline())  
 line = fread.readline()  
 # восстановление ходов игрока  
 while not (line == "BOT\n"):  
 X = int(line)  
 Y = int(fread.readline())  
 HUMAN.append((X, Y))  
 line = fread.readline()  
 line = fread.readline()  
 # восстановление ходов бота  
 while not (line == "END\n"):  
 X = int(line)  
 Y = int(fread.readline())  
 BOT.append((X, Y))  
 line = fread.readline()  
 fread.close()  
 # отрисовка камней игрока  
 i = 0  
 while i < len(HUMAN):  
 screen.blit(user, HUMAN[i])  
 pygame.display.update()  
 i = i + 1  
 # отрисовка камней бота  
 i = 0  
 while i < len(BOT):  
 screen.blit(ai, BOT[i])  
 pygame.display.update()  
 i = i + 1  
 # игровой цикл  
 complete = False  
 while True:  
 pos = [0, 0]  
 # получение события клика  
 for event in pygame.event.get():  
 # выход из игры, когда пользователь нажимает кнопку "Закрыть"  
 if event.type == QUIT:  
 pygame.quit()  
 sys.exit()  
 # работа с дисплеем, когда он свернут  
 if event.type == ACTIVEEVENT or event.type == 17:  
 i = 0  
 while i < len(HUMAN):  
 screen.blit(user, HUMAN[i])  
 pygame.display.update()  
 i = i + 1  
 i = 0  
 while i < len(BOT):  
 screen.blit(ai, BOT[i])  
 pygame.display.update()  
 i = i + 1  
 # определение позиции клика игрока  
 if event.type == MOUSEBUTTONDOWN:  
 pos = list(event.pos)  
 flag = 0  
 # позиция кнопки "RENJU"  
 if pos[0] > 640 and pos[0] < 960 and pos[1] > 0 and pos[1] < 23:  
 screen.blit(about, (640, 87))  
 i = 0  
 while i < len(HUMAN):  
 screen.blit(user, HUMAN[i])  
 pygame.display.update()  
 i = i + 1  
 i = 0  
 while i < len(BOT):  
 screen.blit(ai, BOT[i])  
 pygame.display.update()  
 i = i + 1  
 # перезапуск текущей игры  
 if pos[0] > 670 and pos[0] < 760 and pos[1] > 500 and pos[1] < 590:  
 del (HUMAN)  
 del (BOT)  
 mode1(u, enabled, 1)  
 # выход в меню выбора  
 if pos[0] > 790 and pos[0] < 940 and pos[1] > 400 and pos[1] < 476:  
 del (HUMAN)  
 del (BOT)  
 pygame.quit()  
 select()  
 # сохранение и выход из игры  
 if pos[0] > 790 and pos[0] < 940 and pos[1] > 500 and pos[1] < 590:  
 if not complete:  
 # сохранение параметров игры  
 fwrite = open('info\savedGame.txt', 'w')  
 fwrite.write(str(u) + "\n")  
 fwrite.write(str(enabled) + "\n")  
 fwrite.write(str(count) + "\n")  
 # сохранение ходов игрока  
 i = 0  
 while i < len(HUMAN):  
 a = list(HUMAN[i])  
 fwrite.writelines(str(a[0]) + "\n" + str(a[1]) + "\n")  
 i = i + 1  
 # сохранение ходов бота  
 i = 0  
 fwrite.write("BOT\n")  
 while i < len(BOT):  
 a = list(BOT[i])  
 fwrite.writelines(str(a[0]) + "\n" + str(a[1]) + "\n")  
 i = i + 1  
 fwrite.write("END\n")  
 fwrite.close()  
 pygame.quit()  
 sys.exit()  
 # выход из игры без сохранения  
 elif complete:  
 pygame.quit()  
 sys.exit()  
 # отмена хода игрока за белые или черные камни  
 if (count % 2 == 1 and u == 0) or (count % 2 == 0 and u == 1):  
 # позиция кнопки "отмена хода"  
 if pos[0] > 640 and pos[0] < 960 and pos[1] > 25 and pos[1] < 90 and count > 2:  
 # если отмена хода доступна и игра не завершена, то  
 if undo == 1 and not complete and enabled:  
 undo = 0  
 del (HUMAN[len(HUMAN) - 1])  
 del (BOT[len(BOT) - 1])  
 count = count - 2  
 i = 0  
 while i < len(HUMAN):  
 screen.blit(user, HUMAN[i])  
 pygame.display.update()  
 i = i + 1  
 i = 0  
 while i < len(BOT):  
 screen.blit(ai, BOT[i])  
 pygame.display.update()  
 i = i + 1  
 # ход игрока  
 if pos[0] > 20 and pos[0] < 620 and pos[1] > 20 and pos[1] < 620 and not complete:  
 flag = 1  
 # определение места (центра), в котором должны быть размещены камни  
 if flag == 1:  
 X = pos[0] - pos[0] % 40  
 if pos[0] % 40 > 40 - pos[0] % 40:  
 X = X + 40  
 X = X - 20  
 Y = pos[1] - pos[1] % 40  
 if pos[1] % 40 > 40 - pos[1] % 40:  
 Y = Y + 40  
 Y = Y - 20  
 # проверка доступности хода  
 j = 0  
 while j < len(HUMAN):  
 if X == HUMAN[j][0] and Y == HUMAN[j][1]:  
 flag = 0  
 break  
 j = j + 1  
 j = 0  
 while j < len(BOT):  
 if X == BOT[j][0] and Y == BOT[j][1]:  
 flag = 0  
 break  
 j = j + 1  
 # сохранение последнего хода  
 if flag == 1:  
 undo = 1  
 screen.blit(me, (650, 400))  
 HUMAN.append((X, Y))  
 i = 0  
 # прорисовка ходов игрока  
 while i < len(HUMAN):  
 screen.blit(user, HUMAN[i])  
 pygame.display.update()  
 i = i + 1  
 # прорисовка ходов бота  
 i = 0  
 while i < len(BOT):  
 screen.blit(ai, BOT[i])  
 pygame.display.update()  
 i = i + 1  
 # проверка пяти камней в ряд после каждого шага для игрока  
 temp = 0  
 # горизонтально  
 if (X + 40, Y) in HUMAN and (X + 80, Y) in HUMAN and (X + 120, Y) in HUMAN and (  
 X + 160, Y) in HUMAN:  
 temp = 1  
 elif (X - 40, Y) in HUMAN and (X - 80, Y) in HUMAN and (X - 120, Y) in HUMAN and (  
 X - 160, Y) in HUMAN:  
 temp = 1  
 # вертикально  
 elif (X, Y + 40) in HUMAN and (X, Y + 80) in HUMAN and (X, Y + 120) in HUMAN and (  
 X, Y + 160) in HUMAN:  
 temp = 1  
 elif (X, Y - 40) in HUMAN and (X, Y - 80) in HUMAN and (X, Y - 120) in HUMAN and (  
 X, Y - 160) in HUMAN:  
 temp = 1  
 # диагонали вниз-вправо вверх-влево  
 elif (X + 40, Y + 40) in HUMAN and (X + 80, Y + 80) in HUMAN and (  
 X + 120, Y + 120) in HUMAN and (  
 X + 160, Y + 160) in HUMAN:  
 temp = 1  
 elif (X - 40, Y - 40) in HUMAN and (X - 80, Y - 80) in HUMAN and (  
 X - 120, Y - 120) in HUMAN and (  
 X - 160, Y - 160) in HUMAN:  
 temp = 1  
 # диагональ вверх-вправо вниз-влево  
 elif (X - 40, Y + 40) in HUMAN and (X - 80, Y + 80) in HUMAN and (  
 X - 120, Y + 120) in HUMAN and (  
 X - 160, Y + 160) in HUMAN:  
 temp = 1  
 elif (X + 40, Y - 40) in HUMAN and (X + 80, Y - 80) in HUMAN and (  
 X + 120, Y - 120) in HUMAN and (  
 X + 160, Y - 160) in HUMAN:  
 temp = 1  
 # горизонтально 1 к 3, 2 к 2  
 elif (X - 40, Y) in HUMAN and (X + 40, Y) in HUMAN and (X + 80, Y) in HUMAN and (  
 X + 120, Y) in HUMAN:  
 temp = 1  
 elif (X - 80, Y) in HUMAN and (X - 40, Y) in HUMAN and (X + 40, Y) in HUMAN and (  
 X + 80, Y) in HUMAN:  
 temp = 1  
 elif (X - 120, Y) in HUMAN and (X - 80, Y) in HUMAN and (X - 40, Y) in HUMAN and (  
 X + 40, Y) in HUMAN:  
 temp = 1  
 # вертикально 1 к 3, 2 к 2  
 elif (X, Y - 40) in HUMAN and (X, Y + 40) in HUMAN and (X, Y + 80) in HUMAN and (  
 X, Y + 120) in HUMAN:  
 temp = 1  
 elif (X, Y - 80) in HUMAN and (X, Y - 40) in HUMAN and (X, Y + 40) in HUMAN and (  
 X, Y + 80) in HUMAN:  
 temp = 1  
 elif (X, Y - 120) in HUMAN and (X, Y - 80) in HUMAN and (X, Y - 40) in HUMAN and (  
 X, Y + 40) in HUMAN:  
 temp = 1  
 # диагонали вниз-вправо вверх-влево  
 elif (X - 40, Y - 40) in HUMAN and (X + 40, Y + 40) in HUMAN and (X + 80, Y + 80) in HUMAN and (  
 X + 120, Y + 120) in HUMAN:  
 temp = 1  
 elif (X - 80, Y - 80) in HUMAN and (X - 40, Y - 40) in HUMAN and (X + 40, Y + 40) in HUMAN and (  
 X + 80, Y + 80) in HUMAN:  
 temp = 1  
 elif (X - 120, Y - 120) in HUMAN and (X - 80, Y - 80) in HUMAN and (  
 X - 40, Y - 40) in HUMAN and (X + 40, Y + 40) in HUMAN:  
 temp = 1  
 # диагональ вверх-вправо вниз-влево  
 elif (X - 40, Y + 40) in HUMAN and (X + 40, Y - 40) in HUMAN and (X + 80, Y - 80) in HUMAN and (  
 X + 120, Y - 120) in HUMAN:  
 temp = 1  
 elif (X - 80, Y + 80) in HUMAN and (X - 40, Y + 40) in HUMAN and (X + 40, Y - 40) in HUMAN and (  
 X + 80, Y - 80) in HUMAN:  
 temp = 1  
 elif (X - 120, Y + 120) in HUMAN and (X - 80, Y + 80) in HUMAN and (  
 X - 40, Y + 40) in HUMAN and (  
 X + 40, Y - 40) in HUMAN:  
 temp = 1  
 # объявление победителя  
 if temp == 1:  
 screen.blit(nothing, (650, 400))  
 screen.blit(win, (640, 87))  
 screen.blit(escape, (790, 500))  
 screen.blit(playAgain, (670, 500))  
 i = 0  
 while i < len(HUMAN):  
 screen.blit(user, HUMAN[i])  
 pygame.display.update()  
 i = i + 1  
 i = 0  
 while i < len(BOT):  
 screen.blit(ai, BOT[i])  
 pygame.display.update()  
 i = i + 1  
 pygame.display.update()  
 complete = True  
 count = count + 1  
 # ход бота  
 if ((count % 2 == 0 and u == 0) or (count % 2 == 1 and u == 1)) and not complete:  
 # алгоритм для ии  
 algo.attack(BOT, HUMAN, 0, 6, -1000, 1000)  
 screen.blit(you, (650, 400))  
 pygame.display.update()  
 i = 0  
 while i < len(HUMAN):  
 screen.blit(user, HUMAN[i])  
 pygame.display.update()  
 i = i + 1  
 i = 0  
 while i < len(BOT):  
 screen.blit(ai, BOT[i])  
 pygame.display.update()  
 i = i + 1  
 X = BOT[len(BOT) - 1][0]  
 Y = BOT[len(BOT) - 1][1]  
 # проверка пяти камней в ряд после каждого шага для бота  
 temp = 0  
 # горизонтально  
 if (X + 40, Y) in BOT and (X + 80, Y) in BOT and (X + 120, Y) in BOT and (X + 160, Y) in BOT:  
 temp = 1  
 elif (X - 40, Y) in BOT and (X - 80, Y) in BOT and (X - 120, Y) in BOT and (X - 160, Y) in BOT:  
 temp = 1  
 # вертикально  
 elif (X, Y + 40) in BOT and (X, Y + 80) in BOT and (X, Y + 120) in BOT and (X, Y + 160) in BOT:  
 temp = 1  
 elif (X, Y - 40) in BOT and (X, Y - 80) in BOT and (X, Y - 120) in BOT and (X, Y - 160) in BOT:  
 temp = 1  
 # диагонали вниз-вправо вверх-влево  
 elif (X + 40, Y + 40) in BOT and (X + 80, Y + 80) in BOT and (X + 120, Y + 120) in BOT and (  
 X + 160, Y + 160) in BOT:  
 temp = 1  
 elif (X - 40, Y - 40) in BOT and (X - 80, Y - 80) in BOT and (X - 120, Y - 120) in BOT and (  
 X - 160, Y - 160) in BOT:  
 temp = 1  
 # диагональ вверх-вправо вниз-влево  
 elif (X - 40, Y + 40) in BOT and (X - 80, Y + 80) in BOT and (X - 120, Y + 120) in BOT and (  
 X - 160, Y + 160) in BOT:  
 temp = 1  
 elif (X + 40, Y - 40) in BOT and (X + 80, Y - 80) in BOT and (X + 120, Y - 120) in BOT and (  
 X + 160, Y - 160) in BOT:  
 temp = 1  
 # горизонтально 1к3 и 2к2  
 elif (X - 40, Y) in BOT and (X + 40, Y) in BOT and (X + 80, Y) in BOT and (X + 120, Y) in BOT:  
 temp = 1  
 elif (X - 80, Y) in BOT and (X - 40, Y) in BOT and (X + 40, Y) in BOT and (X + 80, Y) in BOT:  
 temp = 1  
 elif (X - 120, Y) in BOT and (X - 80, Y) in BOT and (X - 40, Y) in BOT and (X + 40, Y) in BOT:  
 temp = 1  
 # # вертикально 1к3 и 2к2  
 elif (X, Y - 40) in BOT and (X, Y + 40) in BOT and (X, Y + 80) in BOT and (X, Y + 120) in BOT:  
 temp = 1  
 elif (X, Y - 80) in BOT and (X, Y - 40) in BOT and (X, Y + 40) in BOT and (X, Y + 80) in BOT:  
 temp = 1  
 elif (X, Y - 120) in BOT and (X, Y - 80) in BOT and (X, Y - 40) in BOT and (X, Y + 40) in BOT:  
 temp = 1  
 # диагонали вниз-вправо вверх-влево  
 elif (X - 40, Y - 40) in BOT and (X + 40, Y + 40) in BOT and (X + 80, Y + 80) in BOT and (  
 X + 120, Y + 120) in BOT:  
 temp = 1  
 elif (X - 80, Y - 80) in BOT and (X - 40, Y - 40) in BOT and (X + 40, Y + 40) in BOT and (  
 X + 80, Y + 80) in BOT:  
 temp = 1  
 elif (X - 120, Y - 120) in BOT and (X - 80, Y - 80) in BOT and (X - 40, Y - 40) in BOT and (  
 X + 40, Y + 40) in BOT:  
 temp = 1  
 # диагональ вверх-вправо вниз-влево  
 elif (X - 40, Y + 40) in BOT and (X + 40, Y - 40) in BOT and (X + 80, Y - 80) in BOT and (  
 X + 120, Y - 120) in BOT:  
 temp = 1  
 elif (X - 80, Y + 80) in BOT and (X - 40, Y + 40) in BOT and (X + 40, Y - 40) in BOT and (  
 X + 80, Y - 80) in BOT:  
 temp = 1  
 elif (X - 120, Y + 120) in BOT and (X - 80, Y + 80) in BOT and (X - 40, Y + 40) in BOT and (  
 X + 40, Y - 40) in BOT:  
 temp = 1  
 # объявление победителя бота  
 if temp == 1:  
 screen.blit(nothing, (650, 400))  
 screen.blit(lost, (640, 87))  
 screen.blit(escape, (790, 500))  
 screen.blit(playAgain, (670, 500))  
 i = 0  
 while i < len(HUMAN):  
 screen.blit(user, HUMAN[i])  
 pygame.display.update()  
 i = i + 1  
 i = 0  
 while i < len(BOT):  
 screen.blit(ai, BOT[i])  
 pygame.display.update()  
 i = i + 1  
 pygame.display.update()  
 complete = True  
 count = count + 1  
 screen.blit(background, (0, 0))  
  
# окно выбора параметров игры  
def select():  
 user = 0  
 undo = 0  
 pygame.init()  
 OPTIONS = "images\select.png"  
 WINDOW1 = pygame.display.set\_mode((660, 390))  
 pygame.display.set\_caption("RENJU")  
 icon = pygame.image.load(i\_icon)  
 options = pygame.image.load(OPTIONS).convert\_alpha()  
 WINDOW1.blit(options, (0, 0))  
 pygame.display.set\_icon(icon)  
 while True:  
 pos = [0, 0]  
 for event in pygame.event.get():  
 if event.type == QUIT:  
 pygame.quit()  
 sys.exit()  
 if event.type == MOUSEBUTTONDOWN:  
 pos = list(event.pos)  
 # выбор цвета камня  
 if pos[1] < 318 and pos[1] > 0:  
 if pos[0] < 330 and pos[0] > 0:  
 user = 1  
 if pos[0] < 660 and pos[0] > 330:  
 user = 0  
 if pos[1] < 360 and pos[1] > 318:  
 # начать игру  
 if pos[0] < 660 and pos[0] > 440:  
 pygame.quit()  
 mode1(user, undo, 1)  
 # возможность отмены хода  
 if pos[0] < 440 and pos[0] > 220:  
 undo = 1  
 # смена темы игрового поля  
 if pos[0] < 220 and pos[0] > 0:  
 fread = open('info\\theme.txt', 'r')  
 prev = fread.readline()  
 fread.close()  
 img = list(prev)  
 img[4] = str((int(img[4]) + 1) % 7)  
 fwrite = open('info\\theme.txt', 'w')  
 i = 0  
 while i < len(img):  
 fwrite.writelines(img[i])  
 i = i + 1  
 fwrite.close()  
 pygame.display.update()  
  
# инициализация игры  
def play\_init():  
 i\_icon = "images\icon.png"  
 mode = "images\selectmode.jpg"  
 pygame.init()  
 WINDOW1 = pygame.display.set\_mode((660, 390))  
 pygame.display.set\_caption("RENJU")  
 icon = pygame.image.load(i\_icon)  
 background = pygame.image.load(mode).convert()  
 WINDOW1.blit(background, (0, 0))  
 pygame.display.set\_icon(icon)  
 while True:  
 pos = [0, 0]  
 for event in pygame.event.get():  
 if event.type == QUIT:  
 pygame.quit()  
 sys.exit()  
 if event.type == MOUSEBUTTONDOWN:  
 pos = list(event.pos)  
 # новая игра  
 if pos[1] < 370 and pos[1] > 20:  
 if pos[0] < 320 and pos[0] > 20:  
 select()  
 # возобновить последнюю сохраненную игру  
 if pos[0] > 340 and pos[0] < 660 and pos[1] > 0 and pos[1] < 100:  
 fread = open('info\savedGame.txt', 'r')  
 u = int(fread.readline())  
 e = int(fread.readline())  
 fread.close()  
 mode1(u, e, 2)  
 pygame.display.update()