**Министерство образования и науки Республики Казахстана**

**Евразийский национальный университет им. Л.Н.Гумилёва**

**Факультет информационных технологий**

**Кафедра “Информационные системы”**

**ОТЧЁТ**

**по учебной практике**

Дата начала практики “22” мая 2023 г.

Дата окончания практики “10” июня 2023 г.b

студента Ермолаева Т., группы ИС-22-02

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(подпись)

Руководитель практики от кафедры: Phd, и.о. профессора Жукабаева Т.К.

(учёная степень, учёное звание, Ф.И.О.) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(подпись)

**Астана, 2023 г.**

Вид практики: Учебная

Тема практики: Разработка игрового приложения на языке программирования Python

Сроки прохождения:

с “22” мая 2023 г. по “10” июня 2023 г.

Цель практики:

Получение теоретических знаний, академических умений, практических навыков разработки игровых приложений, ознакомление с процессом разработки игрового приложения, развитие навыков программирования на языке программирования Python, овладение технологиями и инструментами, получение практического опыта.

В ходе проведения учебной практики студент решает следующие задачи:

1. Изучение игровых концепций и механик;
2. Исследование библиотеки или фреймворка для игровой разработки;
3. Программирование игрового приложения;
4. Тестирование и откладка;
5. Создание пользовательского интерфейса;
6. Документирование процесса разработки;
7. Презентация и демонстрация игрового приложения.

Практика ориентирована на формирование следующих компетенций:

* программирование на языке программирования Python;
* разработка игрового приложения;
* использование библиотек и фреймворков;
* творческое мышление и проблемное решение;
* работа в команде и коммуникация;
* аналитические мышление и самоорганизация.

Содержание

1. [Введение 4](#_Toc136514805)

1.1 Цель практики 4

[1.2 Обзор игрового приложения 4](#_Toc136514811)

[2.Техническое задание [6](#_Toc136514811)](#_Toc136514806)

[2.1 Понятие и история об компьютерной игре. 6](#_Toc136514807)

[2.2 Описание требований и функциональности игрового приложения 10](#_Toc136514808)

[2.3 Используемые технологии и инструменты 11](#_Toc136514808)

[3. Архитектура игрового приложения 12](#_Toc136514809)

[3.1 Обзор архитектурного решения 12](#_Toc136514810)

[3.2 Диаграмма классов(UML-диаграмма) 12](#_Toc136514811)

4. Реализация [18](#_Toc136514812)

[4.1 Описание процесса разработки 18](#_Toc136514813)

[4.2 Примеры кода 20](#_Toc136514814)

[5.Тестирование и откладка игрового приложения 22](#_Toc136514815)

[5.1 Описание проведённых тестов 22](#_Toc136514816)

[5.2 Проблемы и исправления 22](#_Toc136514816)

6. Результаты [24](#_Toc136514812)

6.1 [Оценка выполнения поставленных задач и достижения целей практики 24](#_Toc136514808)

[6.2 Выводы. 28](#_Toc136514807)

[Заключение 30](#_Toc136514817)

[Список литературы. 31](#_Toc136514818)

## **1. Введение**

## **1.1 Цель практики**

Целью данного отчёта является представление результатов учебной практики, выполненной в рамках разработки игрового приложения на языке программирования Python. В течение практики было проведено исследование и разработка игрового приложения с использованием современных технологий и инструментов.

Общая цель практики заключалась в приобретении практических навыков в разработке игровых приложений и применении языка программирования Python для создания интерактивной и увлекательной игровой среды. В процессе практики ставились следующие задачи:

1. Анализ требований: изучение основных функциональных и нефункциональных требований к игровому приложению.
2. Проектирование архитектуры: разработка структуры приложения и определение основных компонентов и модулей.
3. Реализация: создание игрового приложения на языке программирования Python с использованием соответствующих библиотек и инструментов.
4. Тестирование и отладка: проведение тестов для проверки функциональности и корректности работы приложения.
5. Оценка результатов и выводы: анализ выполненной работы, оценка достигнутых результатов и формулирование выводов по учебной практике.

В данном отчёте будет представлена детальная информация о разработанном игровом приложении, описан процесс его создания, а также будут рассмотрены результаты тестирования и проведенного исследования. Этот отчёт предоставит полное представление о выполненной работе и полученных навыках в области разработки игровых приложений с использованием языка программирования Python.

**1.2 Обзор игрового приложения**

В качестве объекта разработки было выбрано игровое приложение "Тетрис". Это популярная логическая игра, в которой игроку необходимо управлять падающими тетромино-фигурами и формировать горизонтальные линии. Когда заполняется полная горизонтальная линия, она исчезает, освобождая место для новых фигур. Цель игры состоит в том, чтобы набрать максимальное количество очков, не допуская заполнения игрового поля.

Основные особенности и функциональность игрового приложения "Тетрис" включают:

1. Управление фигурами: игрок может поворачивать, перемещать и ускорять падение тетромино-фигур с помощью клавиш клавиатуры.
2. Игровое поле: игра происходит на прямоугольном игровом поле, где фигуры падают сверху вниз.
3. Очки и уровни: игра отслеживает количество заполненных линий и предоставляет игроку очки за каждую удаленную линию. При достижении определенного количества очков уровень игры повышается.
4. Система ускорения: игрок имеет возможность ускорять падение фигур, чтобы лучше контролировать их расположение на игровом поле.
5. Графическое оформление: игровое приложение имеет привлекательный и понятный интерфейс с графическим отображением фигур и игрового поля.

В ходе практики была выполнена разработка и реализация игрового приложения "Тетрис" с использованием языка программирования Python и соответствующих библиотек и инструментов. В следующих разделах отчёта будет более подробно рассмотрено проектирование и реализация игрового приложения "Тетрис".

**2. Техническое задание**

## **2.1 Понятие и история о компьютерной игре**

Компьютерная игра - это вид развлечения, основанный на использовании компьютерной технологии, где игрок взаимодействует с виртуальным миром, управляет персонажами и выполняет задачи, целью которых часто является достижение победы, завершение заданий или просто получение удовольствия от игрового процесса.

Компьютерные игры могут быть различных жанров, таких как приключенческие, стратегические, ролевые, шутеры от первого и третьего лица, спортивные, головоломки и многие другие. Они могут быть разработаны как для одиночной игры, так и для многопользовательского режима, где игроки могут соревноваться или сотрудничать с другими игроками через Интернет. Основными компонентами компьютерных игр являются игровой движок (программное обеспечение, отвечающее за отображение графики, звуковые эффекты, физику и взаимодействие с игровым миром), искусственный интеллект (программное обеспечение, управляющее поведением виртуальных персонажей и противников) и игровой сценарий (сюжет, задания, цели, миссии и другие элементы, которые определяют игровой опыт). Компьютерные игры имеют огромную популярность среди различных возрастных групп и стали важной частью современной популярной культуры. Они предлагают игрокам возможность отдохнуть, расслабиться и провести время в виртуальных мирах, а также развивать навыки, такие как стратегическое мышление, реакцию, координацию движений и командное взаимодействие. Компьютерные игры также используются в образовательных целях, симуляторах, тренировках и других областях, где важно практическое применение навыков и умений.

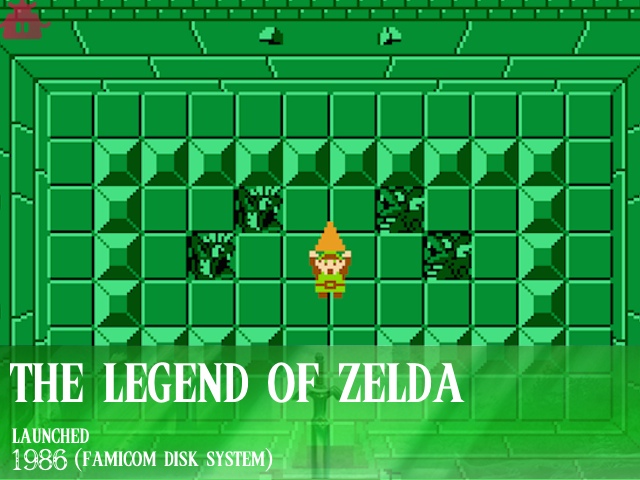
Однажды в далеком 1972 году в маленькой гаражной лаборатории появился первый прототип компьютерной игры. Эта игра называлась "Pong" и представляла собой простую симуляцию теннисного матча, где два игрока могли управлять ракетками, отбивая мяч друг от друга. "Pong" стал первым коммерчески успешным проектом в индустрии компьютерных игр и заложил основу для будущего развития.



*Рисунок 1. Аркадный автомат для игры в Pong, 1972*

С течением времени компьютерные игры продолжали развиваться и превращались во что-то гораздо более сложное и захватывающее. В 1980-х годах появились первые домашние игровые консоли, такие как Atari 2600 и Nintendo Entertainment System (NES), которые позволяли игрокам наслаждаться играми прямо у себя дома. Игры стали становиться более детализированными и разнообразными, от аркадных шутеров до приключенческих игр с глубоким сюжетом.

С появлением персональных компьютеров в 1980-х и 1990-х годах игры стали еще более разнообразными и многоуровневыми. Появились ролевые игры (RPG) такие, как "The Legend of Zelda" и "Final Fantasy", где игроки могли погрузиться в фантастические миры и выполнять сложные задания. Стратегические игры, такие как "Civilization" и "Warcraft", позволяли игрокам управлять виртуальными империями и вести войны.



*Рисунок 2. Геймплей The Legend of Zelda, 1986.*

С развитием графических технологий в конце 1990-х годов появились 3D игры, которые позволяли игрокам полностью погрузиться в виртуальный мир. Компании, такие как id Software с их игрой "Doom", и Valve Corporation с "Half-Life", внесли значительный вклад в развитие игровой индустрии и создали образцы шутеров от первого лица

В 2000-х годах игры стали еще более реалистичными и социальными. Онлайн-игры, такие как "World of Warcraft", позволяли игрокам вступать в виртуальные миры вместе с тысячами других игроков.



# *Рисунок 3. Интерфейс WOW3.*

История Тетриса началась в 1984 году, когда российский программист Алексей Пажитнов разработал игру в свободное время, работая в компании Дородницына Академии наук СССР. Он создал игру, вдохновленную головоломкой Пентамино, в которой игрокам нужно было заполнять горизонтальные линии падающими блоками.

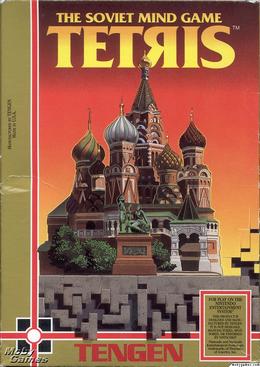
Пажитнов назвал свою игру "Тетрис", объединив слова "тетра" (греческое слово, означающее "четыре") и "теннис" (его любимый вид спорта). Первоначально, игра была разработана для использования на компьютерах серии Electronika 60, распространенных в Советском Союзе.



*Рисунок 4. Tetris (Electronika 60), 1984*

Скоро Тетрис стал популярным среди сотрудников компании и их друзей. Игра пользовалась таким успехом, что она распространилась среди различных компьютерных пользователей в Советском Союзе. В 1986 году Алексей Пажитнов заключил сделку с компанией Elektronorgtechnica (Электроноргтехника), что позволило ему официально выпускать игру.

Вскоре после этого, игра Тетрис привлекла внимание западных разработчиков и издателей компьютерных игр. Компания Atari получила права на выпуск Тетриса для игровых консолей и персональных компьютеров в Северной Америке. Однако, из-за сложностей в отношениях между Atari и руководителями Советского Союза, эта сделка была отменена.



*Рисунок 5. Tetris: The Soviet Mind Game*

Впоследствии, в 1989 году, компания Nintendo выпустила версию Тетриса для своей портативной игровой консоли Game Boy. Игра стала одним из ключевых факторов успеха Game Boy и считается одной из самых продаваемых игр в истории.



*Рисунок 6. Nintendo Gameboy*

С тех пор Тетрис получил множество версий и адаптаций на различные платформы, включая компьютеры, игровые консоли и мобильные устройства. Он стал классикой в мире компьютерных игр и символом популярности головоломок. Тетрис стал неотъемлемой частью игровой культуры и продолжает привлекать миллионы игроков со всего мира, предлагая им захватывающий и увлекательный геймплей.

**2.2 Описание требований и функциональности игрового приложения**

В данном разделе представлено описание требований и функциональности игрового приложения "Тетрис".

Основные требования:

* Игровое приложение должно быть реализовано на языке программирования Python.
* Приложение должно иметь графический интерфейс пользователя (GUI), обеспечивающий удобное взаимодействие игрока с игрой.
* Приложение должно поддерживать отображение игрового поля, падающих тетромино-фигур, текущего счета и уровня игры.
* Приложение должно иметь возможность управления фигурами с помощью клавиш клавиатуры.
* Игра должна предоставлять возможность сохранения и загрузки текущего состояния игры.

Функциональность игрового приложения:

* Отображение игрового поля: приложение должно отображать игровое поле с заданными размерами, где будут падать тетромино-фигуры.
* Управление фигурами: игрок должен иметь возможность поворачивать и перемещать падающие фигуры с помощью клавиш клавиатуры.
* Падение фигур: фигуры должны падать по определенному интервалу времени и останавливаться, если достигнута нижняя граница игрового поля или другие фигуры.
* Заполнение линий: если вся горизонтальная линия заполняется тетромино-фигурами, она должна исчезать, освобождая место для новых фигур.
* Система очков и уровней: игрок должен получать очки за каждую удаленную линию. При достижении определенного количества очков уровень игры должен повышаться, увеличивая скорость падения фигур.
* Конец игры: игра должна заканчиваться, если новая фигура не может быть размещена в верхней части игрового поля.

Дополнительные требования:

* Интерфейс пользователя: игровое приложение должно иметь привлекательный и интуитивно понятный интерфейс, обеспечивающий удобное управление и отображение игровой информации.
* Звуковые эффекты: приложение может содержать звуковые эффекты, например, при падении фигур или удалении линий.
* Режимы игры: приложение может предоставлять различные режимы игры, например, классический режим, режим с ограниченным временем или мультиплеерный режим.
* Выполнение данных требований и функциональности позволит создать полноценное игровое приложение "Тетрис" на языке программирования Python.

**2.3 Используемые технологии и инструменты**

Для разработки игрового приложения "Тетрис" на языке программирования Python будут использованы следующие технологии и инструменты:

1. Язык программирования: Python. Python является интерпретируемым, высокоуровневым языком программирования, который обладает простым синтаксисом и мощными инструментами для разработки.
2. Графическая библиотека: Tkinter. Tkinter - это библиотека, специально разработанная для создания компьютерных игр на языке Python. Она предоставляет множество функций для работы с графикой, звуком, анимацией и обработкой ввода от пользователя. Tkinter обеспечивает удобный способ создания игрового окна, отображения изображений, рисования графических элементов, обработки событий мыши и клавиатуры, воспроизведения звуковых эффектов и многое другое.
3. Интегрированная среда разработки (IDE): Для разработки приложения можно использовать различные IDE, такие как PyCharm, Visual Studio Code, Spyder и другие. Инструменты IDE обеспечивают удобную среду разработки, подсветку синтаксиса, автодополнение кода, отладку и другие полезные функции, которые облегчают процесс разработки.
4. Документация: В процессе разработки приложения создавалась документация, описывающая функциональность, структуру и использование кода. Для документирования можно использовать инструменты, такие как Sphinx или MkDocs. Эти инструменты позволяют создавать профессионально оформленную документацию в виде веб-страниц или PDF-файлов, которая помогает разработчикам и пользователям разобраться в коде и использовании приложения.
5. Тестирование: Для проверки функциональности приложения рекомендуется использовать модульное тестирование с помощью фреймворка, такого как unittest, а также проводить ручное тестирование для проверки взаимодействия игровых компонентов.

Использование данных технологий и инструментов обеспечит эффективную и удобную разработку игрового приложения "Тетрис" на языке программирования Python с помощью библиотеки Tkinter. Эти инструменты обладают обширной функциональностью и поддержкой сообщества, что упрощает создание и поддержку игрового приложения.

# **3. Архитектура игрового приложения**

**3.1 Обзор архитектурного решения**

## Архитектура игрового приложения "Тетрис" может быть организована следующим образом, с использованием различных компонентов и модулей:

## Модуль отображения:

## Отвечает за отображение игрового окна и графических элементов.

## Включает функции для рисования игрового поля, тетромино-фигур, счета и других игровых элементов.

## Отслеживает пользовательский ввод, включая нажатия клавиш клавиатуры для управления фигурами.

1. Модуль игровой логики:

* Обрабатывает основные правила игры, такие как перемещение фигур, их повороты, падение фигур и проверку условий завершения игры.
* Отвечает за обработку столкновений фигур и удаление заполненных горизонтальных линий.
* Управляет счетом игрока, уровнем игры и скоростью падения фигур в зависимости от достижений игрока.

1. Модуль сохранения и загрузки:

* Обеспечивает возможность сохранения текущего состояния игры и загрузки сохраненных состояний.
* Включает функции для сохранения и загрузки данных, таких как положение фигур, счет и уровень игры.

1. Модуль управления:

* Обеспечивает взаимодействие между модулями и компонентами приложения.
* Отвечает за инициализацию игры, запуск основного цикла игры и обработку событий.

1. Модуль статистики и рекордов:

* Отслеживает и хранит статистику игры, такую как общий счет, количество удаленных линий, время игры и другие достижения игрока.
* Включает функции для отображения и обновления статистических данных.
* Может также содержать функциональность для сохранения и отображения рекордов игроков.

1. Модуль звукового сопровождения:

* Отвечает за воспроизведение звуковых эффектов и музыки в игре.
* Воспроизводит звуки при падении фигур, удалении линий и других игровых событиях.

Кроме вышеперечисленных модулей, архитектура игры "Тетрис" может также включать дополнительные компоненты, такие как модуль уровней и сложности, который управляет уровнями игры и регулирует сложность, модуль мультиплеера, который позволяет игрокам играть вместе или соревноваться друг с другом, а также модуль настроек и пользовательского интерфейса, позволяющий игрокам настраивать параметры игры и взаимодействовать с приложением.

## **3.2 Диаграмма классов(UML-диаграмма)**

UML-диаграммы (Unified Modeling Language) используются для визуализации, проектирования и документирования различных аспектов программных систем. Они представляют собой графический язык, который позволяет разработчикам и аналитикам моделировать структуру, поведение, взаимодействие и процессы в системе.

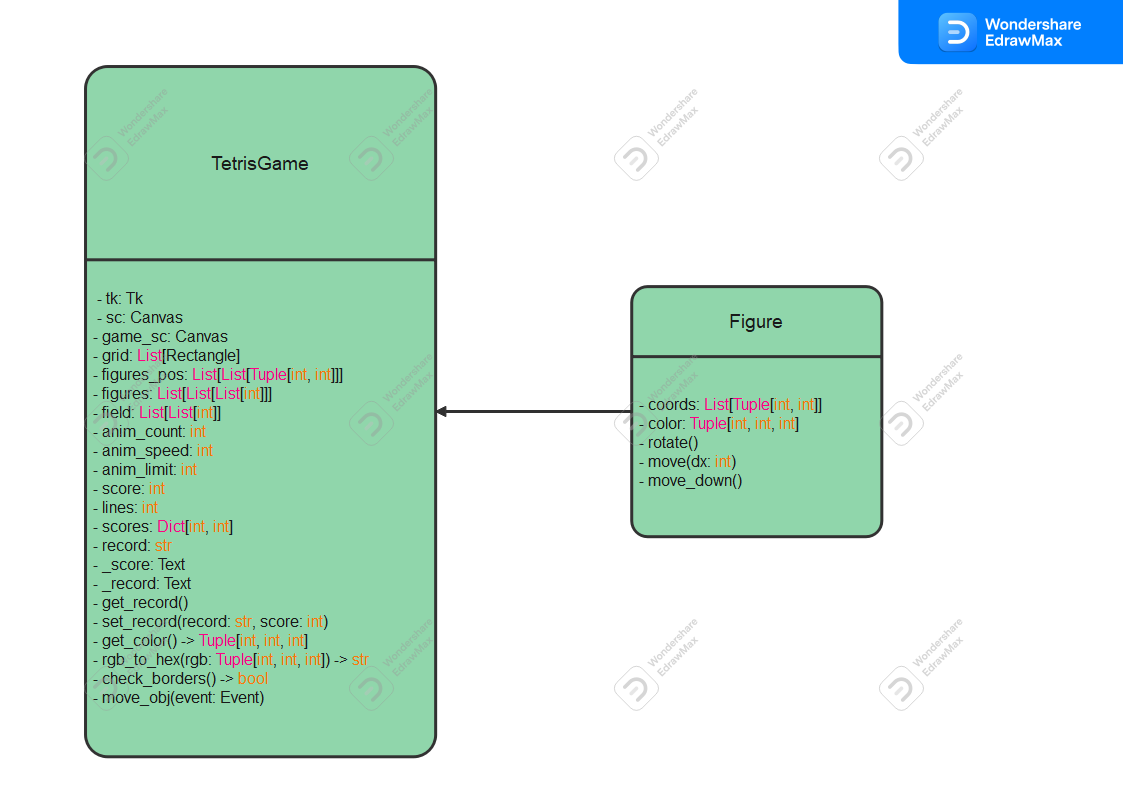
Основные причины использования UML-диаграмм – это визуализация структуры и поведения: UML-диаграммы позволяют визуализировать компоненты системы, их атрибуты, методы, отношения и взаимодействия между ними. Это помогает разработчикам и заинтересованным сторонам лучше понять систему и ее архитектуру.

Также они наглядно отображают процессы и поведение системы: UML-диаграммы позволяют описывать процессы, поведение и последовательность действий в системе. Например, диаграммы последовательности и диаграммы состояний могут показать, как объекты взаимодействуют друг с другом и как меняется их состояние в процессе работы системы.

UML-диаграммы являются универсальным языком, который позволяет команде разработчиков и заинтересованным сторонам обмениваться информацией и идеями. Они создают единое понимание системы и могут служить средством коммуникации между разработчиками, дизайнерами, аналитиками и заказчиками.

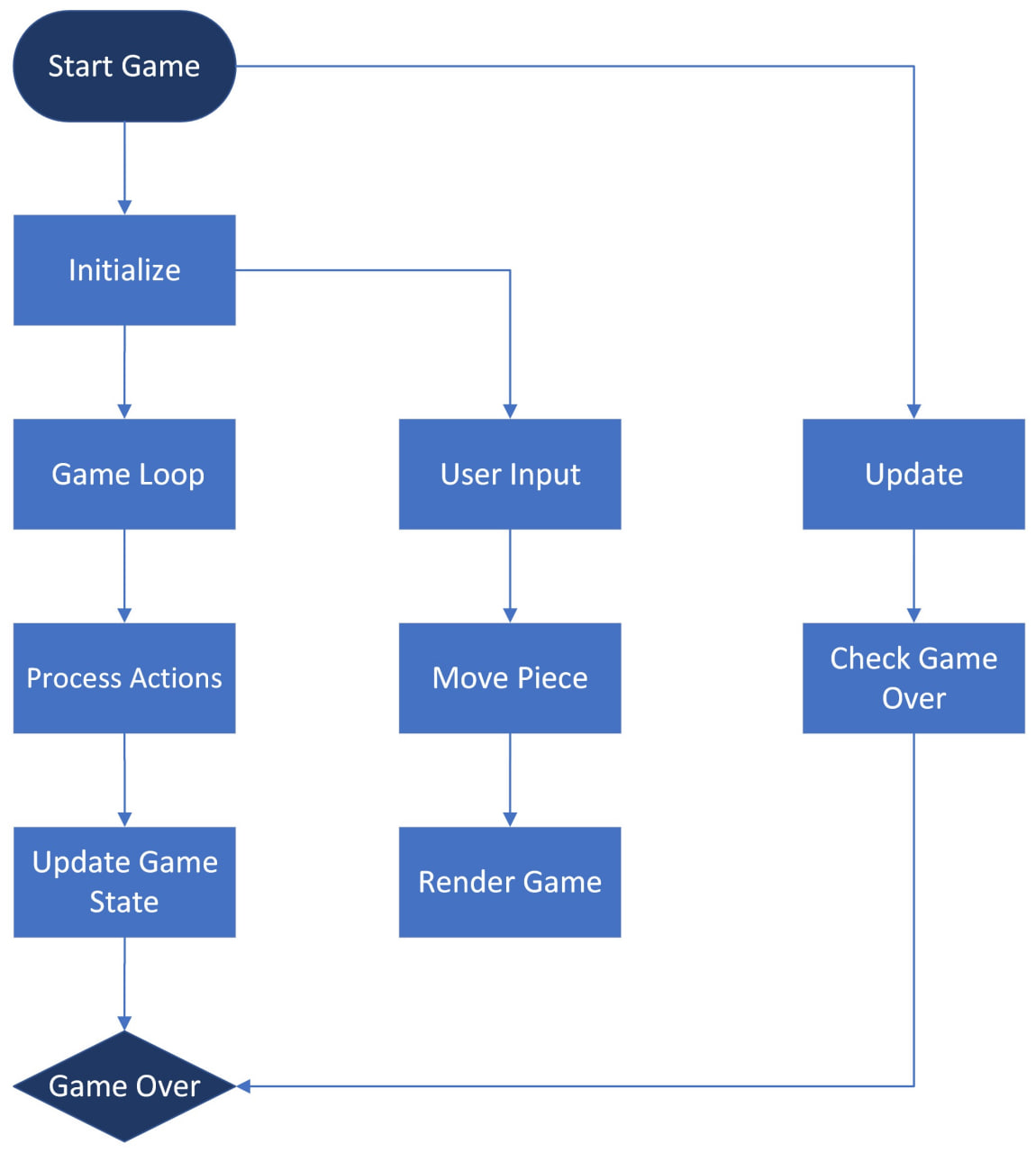
Можно сказать, что UML — это набор правил, по которым нужно рисовать схемы. Зная его, можно быстро создавать универсальные графические представления сложных процессов и структур. Именно поэтому IT-специалисты во время разработки ПО часто используют UML-моделирование и проектирование процессов.

Схему на языке UML можно составить по уже существующему объекту или процессу либо создать на этапе проектирования, чтобы разрабатывать объект или отлаживать процесс. Диаграммы UML применяют в проектировании, презентациях, описании или создании документации. В аналитике данных тоже используют UML — например, чтобы описать аналитическую программу или структуру информации в проекте. На курсе «Аналитик данных» студентов обучают построению и проектированию сложных систем, в том числе на языке UML.

В качестве примера наша команда продемонстрирует несколько образцов UML-диаграмма. Для создания UML-диаграмм была использована такая программа, как Wondershare EdrawMax.

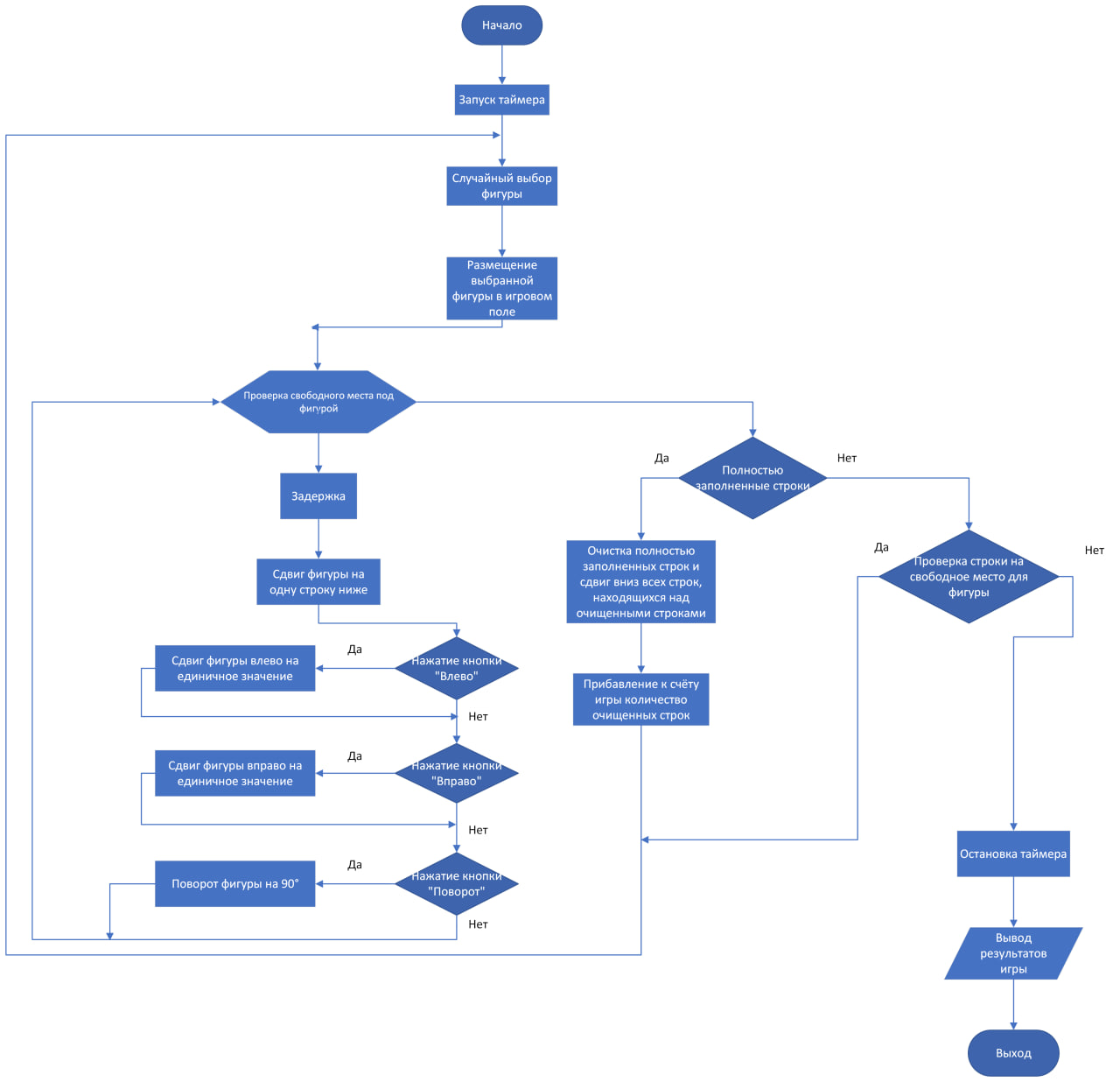
*Рисунок 7. Class diagram*

Диаграмма классов (от англ. "class diagram") - необходима для постановки задачи и её анализу при разработке приложения. В диаграмме классов можно визуально представить структуру программы в виде классов и взаимосвязей между ними. Таким образом в каждой ячейке указывается уникальное название класса, её параметры и методы.



*Рисунок 8. Activity diagram*

Диаграмма деятельности (от англ. "activity diagram") - это графическое представление последовательности действий и потоков управления в системе или процессе. Она является одной из формализованных диаграмм, используемых в моделировании бизнес-процессов, системного анализа и разработки программного обеспечения. Диаграмма деятельности состоит из набора символов и стрелок, которые представляют действия, решения, ветвления и потоки управления. В основе диаграммы лежит идея моделирования процессов в виде последовательности шагов или действий, которые выполняются в системе или приложении.



*Рисунок 9. Flowchart*

Блок-схема (от англ. “flowchart”) — это графическое представление последовательности операций, процессов или алгоритмов с использованием блоков различной формы, соединенных стрелками, указывающими направление потока выполнения. Она широко используется в области программирования, системного анализа и проектирования, а также в других областях, где необходимо визуализировать последовательность действий или процессов.

Важно отметить, что блок-схема не является строгими инструкциями по выполнению операций, а скорее представляет общую структуру и поток алгоритма или процесса. Она помогает визуализировать и анализировать операции и их последовательность, что делает ее полезным инструментом для моделирования и проектирования различных видов систем и процессов.

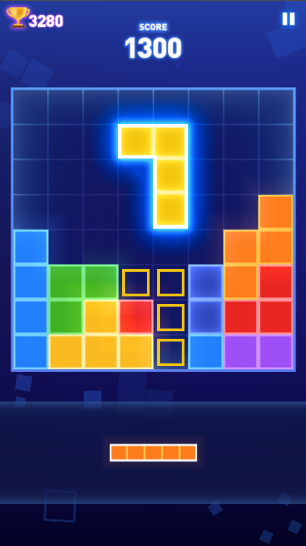
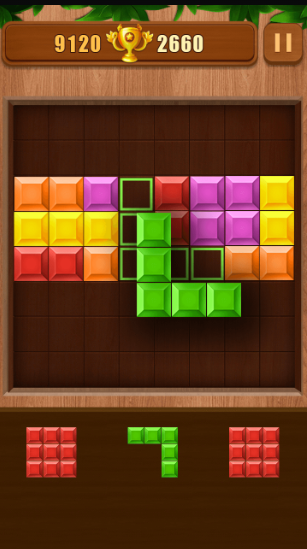
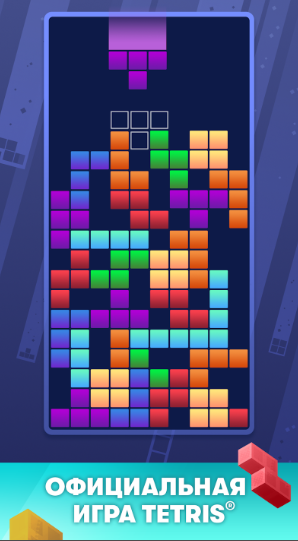
**4. Реализация**

**4.1 Описание процесса разработки**

Следующим этапом является создание игрового приложения, то есть написание программного кода. Для написания кода использовался такой язык программирования, как Python.

Python — высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нём программ. Язык является полностью объектно-ориентированным в том плане, что всё является объектами. Необычной особенностью языка является выделение блоков кода пробельными отступам. Синтаксис ядра языка минималистичен, за счёт чего на практике редко возникает необходимость обращаться к документации. Сам же язык известен как интерпретируемый и используется в том числе для написания скриптов. Недостатками языка являются зачастую более низкая скорость работы и более высокое потребление памяти написанных на нём программ по сравнению с аналогичным кодом, написанным на компилируемых языках, таких как C или C++ и Python предназначен для анализа, хранения, обработки данных и не используется для созданий игровых приложений, однако благодаря использованию такой библиотеки, как “tkinter” позволяет реализовать игровое приложение.

Очевидно, что Tetris далеко не наша идея, а потому при создании своей игры, не лишним будет посмотреть на уже готовые реализации этой легендарной головоломки. Так как тетрис на ПК уже давно отошел от канонов и каждый проект пытается разбавить свою игру новыми сложными механиками, а мы больше основываемся на классической версии, мы обратим свой взор на мобильный гейминг.



*Рисунок 10. 3 различных клона Тетриса из App Store и Google Play*

Загрузив и поиграв в 3 различных клона тетриса были сделаны определенные выводы. Прежде всего были найден базис игры. Это полная рандомизация фигур, отслеживание счета с помощью кол-ва заполненных горизонтальных линий, средняя скорость падения фигур, а также показ следующей фигуры, до падения текущей.

В процессе разработки игрового приложения "Тетрис" мы начали с импорта необходимых модулей, таких как tkinter для создания графического интерфейса, messagebox для вывода диалоговых окон, random для генерации случайных значений, copy для создания копий объектов и time для задержки обновления экрана.

Затем мы приступили к настройке графического интерфейса. Создали главное окно приложения (Tk) и настроили его параметры, включая заголовок окна, размеры и настройки отображения. Создали холсты (Canvas), которые использовались для отображения игрового поля и других элементов интерфейса.

Далее мы загрузили необходимые ресурсы для приложения. Загрузили изображения, такие как фоновое изображение и изображение следующей фигуры, и установили их на соответствующие холсты. Также создали сетку на игровом поле, которая представляла собой набор прямоугольников, отображающих ячейки поля.

Для работы с фигурами в игре мы определили список возможных позиций и форм фигур. Каждая фигура представлена набором координат ячеек на игровом поле. Мы создали функции для проверки границ поля и возможности перемещения и поворота фигуры.

Далее реализовали основной игровой цикл. В этом цикле происходило обновление положения текущей фигуры, проверка коллизий, заполнение строк и подсчет очков. Мы также обрабатывали пользовательский ввод, позволяя перемещать и поворачивать фигуры.

Во время игры мы отслеживали текущий счет, количество заполненных строк и рекорд игры. Рекорд сохранялся в файле, и мы использовали функции для его чтения и записи.

Если игрок достигал верхней границы поля, игра считалась оконченной. Мы обновляли рекорд, если текущий счет превышал сохраненный рекорд, и сбрасывали поле и счет в начальное состояние.

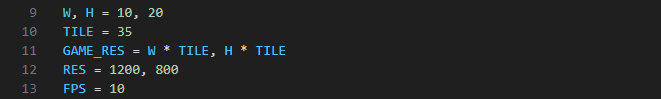
В конце игры мы уничтожали графический интерфейс и завершали приложение.

В результате нашего процесса разработки мы создали полноценное игровое приложение "Тетрис" с графическим интерфейсом, возможностью управления фигурами и отображением счета.

Tkinter - это стандартная библиотека языка Python, которая предоставляет инструменты для создания графического интерфейса пользователя (GUI). Она является частью стандартной библиотеки Python и доступна для использования без необходимости установки дополнительных пакетов. Библиотека Tkinter предоставляет различные виджеты (элементы интерфейса) и методы для создания окон, кнопок, текстовых полей, меток и других элементов интерфейса. Она также предлагает различные способы управления размещением элементов на экране, обработки событий, анимации и рисования на холсте. С использованием Tkinter вы можете создавать интерактивные приложения с графическим интерфейсом, которые позволяют пользователям взаимодействовать с программой, вводить данные, получать вывод и управлять ее функциональностью. Она широко используется для создания простых и средних по сложности GUI-приложений, включая игры, утилиты, инструменты администрирования и другие программы с графическим интерфейсом. Tkinter является кросс-платформенным и поддерживается на различных операционных системах, таких как Windows, macOS и Linux. Она имеет простой и интуитивно понятный API, что делает ее доступной для начинающих разработчиков, а также предлагает достаточно гибкие возможности для создания сложных интерфейсов и приложений при необходимости.

В коде Tkinter используется для создания графического интерфейса игрового приложения "Тетрис". Она предоставляет возможность создания оконного приложения, размещения холстов для отображения игрового поля и элементов интерфейса, обработки пользовательского ввода и других функциональностей, необходимых для создания игры.

**4.2 Примеры кода**



*Рисунок 11. Пример кода для установки размера игрового поля, разрешения и частоты кадров.*

Данный код определяет некоторые константы и переменные для управления размерами и разрешением окна игры Tetris.

*W* и *H* определяют ширину и высоту игрового поля в клетках. В данном случае, игровое поле будет иметь размер 10 клеток в ширину и 20 клеток в высоту.

*TILE* определяет размер одной клетки в пикселях. В данном случае, каждая клетка будет иметь размер 35x35 пикселей.

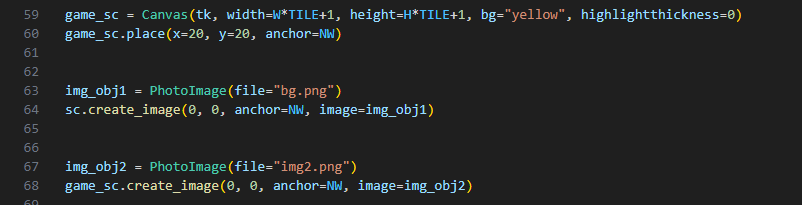
*GAME\_RES* - это кортеж, содержащий размеры окна игры, вычисленные путем умножения ширины (W) и высоты (H) игрового поля на размер клетки (TILE). Например, если W = 10, H = 20, и TILE = 35, то GAME\_RES будет (350, 700).

*RES* определяет размеры окна приложения в пикселях. В данном случае, окно приложения будет иметь размер 1200x800 пикселей.

*FPS* определяет количество кадров в секунду (фреймов в секунду), которые будут отображаться в игре. В данном случае, игра будет обновляться с частотой 10 кадров в секунду.

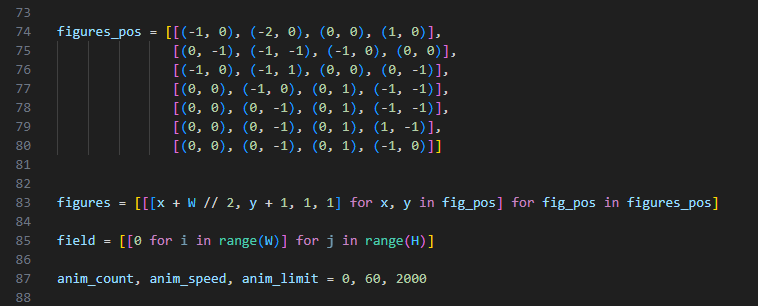
Эти переменные и константы используются для определения размеров игрового поля,

размеров окна игры, а также для управления скоростью и плавностью анимаций в игре Tetris.



*Рисунок 12. Пример работы с изображением в библиотеке Tkinter.*

В данном случае *game\_sc* и *game\_sc.place* создает новый виджет *Canvas* (холст) и размещает его в главном окне *tk* приложения. Холст используется для отображения игрового поля. Далее с помощью команд *game.sc.create\_image* и *sc.create\_image* мы создаем задний фон и фон для игрового поля с помощью файлов bg.png и img2.png, размещенных в папке с файлом tetris.py. Если наши изображения находятся в другой папке, к ним необходимо прописать путь.

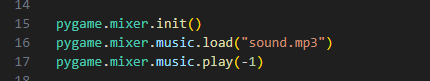


*Рисунок 13. Пример работы создании анимации*

*figures\_pos* представляет собой список, содержащий координаты каждой ячейки, составляющей форму фигуры. Каждая фигура представлена в виде списка координат, где каждый кортеж (-1, 0), (0, -1), (1, 0), и т.д. указывает смещение относительно центральной ячейки фигуры. Например, фигура "L" состоит из четырех ячеек с относительными координатами (-1, 0), (-2, 0), (0, 0), и (1, 0).

*figures* - это список фигур, представленных в виде списка координат. Для каждой фигуры из *figures\_pos* происходит сдвиг координаты центральной ячейки на половину ширины поля «(W // 2)» и на 1 вниз «(y + 1)». Затем создается список координат каждой ячейки фигуры с добавлением смещений. Например, фигура «L» будет иметь список координат [4, 1, 1, 1], [3, 1, 1, 1], [5, 1, 1, 1], [6, 1, 1, 1], где каждый подсписок представляет координаты одной ячейки фигуры.

Таким образом, *figures* содержит все возможные фигуры, которые могут появиться в игре.



*Рисунок 14. Пример работы со звуковыми эффектами и музыкой в библиотеке Pygame*

Вызывается функция *pygame.mixer.init()*, чтобы инициализировать звуковой микшер *Pygame*. Затем загружается файл фоновой музыки с помощью *pygame.mixer.music.load*("sound.mp3"). После чего, фоновая музыка воспроизводится в бесконечном цикле с использованием *pygame.mixer.music.play*(-1), где -1 указывает, что музыка будет воспроизводиться циклично.

**5. Тестирование и откладка игрового приложения**

**5.1 Описание проведённых тестов**

И конечным этапом является тестирование и откладка игрового приложения.

Мой первый шаг был ознакомление с функциональностью игры и ее основными элементами. Я провела функциональное тестирование, чтобы убедиться, что все элементы игры работают правильно. Я проверила, что блоки двигаются и поворачиваются без проблем, заполненные линии удаляются, очки подсчитываются корректно, а игровое поле и интерфейс отображаются правильно.

Вот несколько примеров тестовых случаев:

1. Тестирование пользовательского ввода:

* Тестовый случай: Нажатие клавиши "влево" для перемещения фигуры влево.
* Ожидаемый результат: Фигура сдвигается на одну ячейку влево, не выходя за границы игрового поля.

1. Тестирование счета и записи рекорда:

* Тестовый случай: Заполнение одной или нескольких горизонтальных линий.
* Ожидаемый результат: Заполненные линии исчезают, счет игрока увеличивается и отображается на экране. При достижении нового рекорда рекорд обновляется.

1. Тестирование окончания игры:

* Тестовый случай: Достижение верхней границы игрового поля новой фигурой.
* Ожидаемый результат: Игра завершается, текущий счет сравнивается с рекордом, и при необходимости рекорд обновляется.

1. Тестирование визуальных элементов:

* Тестовый случай: Отображение фигур, фона и текстовых элементов.
* Ожидаемый результат: Фигуры отображаются корректно на игровом поле, фон и текстовые элементы отображаются четко и читаемо.

Данные тестирования были проведены для откладки и устранения различных ошибок, возникших на этапе разработки.

Ошибки исправлялись с помощью итеративного подхода. Каждое исправление проходило через этапы проверки и верификации, чтобы убедиться, что проблема решена и не вызывает новых проблем в других аспектах игры.

После множества тестов и исправлений, я была удовлетворена результатами. Игровое приложение Тетрис было протестировано в разных сценариях, ошибки были обнаружены и исправлены, и игра предлагала стабильный и увлекательный игровой опыт.

**5.2 Проблемы и исправления**

В процессе разработки программного обеспечения невозможно избежать возникновения проблем. Программы становятся все более сложными, требования пользователей могут меняться, а взаимодействие с другими компонентами системы может быть непредсказуемым.

Вот некоторые проблемы, которые возникали на этапе разработки:

1. Фигуры заходили за границы игрового поля.

Описание проблемы: Во время перемещения или вращения фигуры, она выходила за пределы игрового поля, что приводило к некорректному отображению или даже исчезновению фигуры.

Решение: Была добавлена проверка границ перед каждым перемещением или вращением фигуры. Если текущие координаты фигуры выходят за пределы игрового поля, запретить перемещение или вращение в этом направлении.

Например, при перемещении влево мы добавили проверку, что x-координата левой стороны фигуры должна быть больше нуля. Если она меньше или равна нулю, то фигура находится слишком близко к левой границе поля и не должна перемещаться влево.

1. Фигуры могут перекрывать друг друга.

Описание проблемы: Во время перемещения или вращения фигуры, она пересекает или перекрывает уже расположенные фигуры на игровом поле. Это приводило к неправильному отображению или столкновению фигур.

Решение: Была проведена проверка, что при каждом перемещении или вращении фигуры она не пересекается с уже расположенными фигурами. Для этого проверялась каждая ячейка фигуры и соответствующая ячейка игрового поля на перекрытие. Если обнаружено перекрытие, запретить перемещение или вращение фигуры.

Например, можно использовать двумерный массив (матрицу), где каждый элемент представляет ячейку игрового поля. При перемещении или вращении фигуры, проверить соответствующие элементы матрицы на наличие уже занятых ячеек. Если в любой из ячеек фигуры уже есть фигура из матрицы, значит, произошло перекрытие.

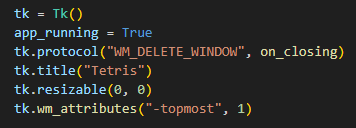
**6. Результаты**

**6.1 Оценка выполнения поставленных задач и достижения целей практики**

Подводя итоги разработки игрового приложения, в течение этого периода был получен ценный опыт и приобретены навыки, которые будут полезны в дальнейшей карьере разработчиков.

В ходе учебной практики приступили к созданию игрового приложения, используя язык программирования Python. Ниже представлена оценка выполнения поставленных задач и достижений целей практики:

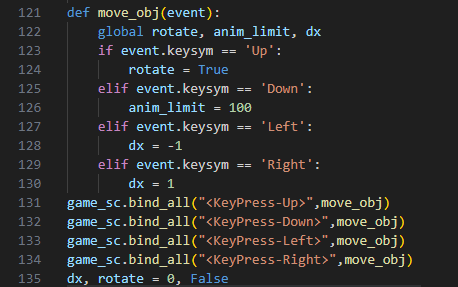
Изучение новых технологий и инструментов: В рамках практики было проведено исследование и изучение новых технологий и инструментов, связанных с разработкой программного обеспечения. Были изучены основные принципы работы, функциональные возможности и способы применения этих технологий. Поставленная цель была достигнута успешно, и полученные знания и навыки могут быть применены в будущих проектах.



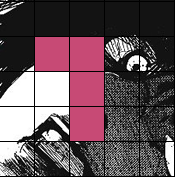
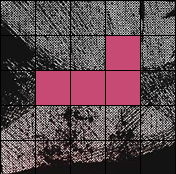


*Рисунок 15. Создание окна игры. В данном случае оно всегда встает поверх других окон. Также заблокирована возможность изменения его размера.*

Разработка программного решения: Была поставлена задача разработать программное решение, решающее определенную проблему или выполняющее определенные функции. В процессе практики были выполнены этапы проектирования, реализации и тестирования программного решения. Результаты разработки соответствуют поставленным требованиям, и программное решение работает корректно и эффективно. Например, управление. При правильной настройке, оно может сделать игровой процесс куда более комфортным и захватывающим, даже в такой простой и однобокой игре как Tetris.



*Рисунок 16. Функция для создания управление. А именно таких действий как, перемещение фигур влево и вправо, их поворот по часовой стрелке, а также их ускоренное падение.*

*Рисунок 17. Поворот фигуры по часовой стрелке, после нажатия на верхнюю стрелочку (клавиша UP).*

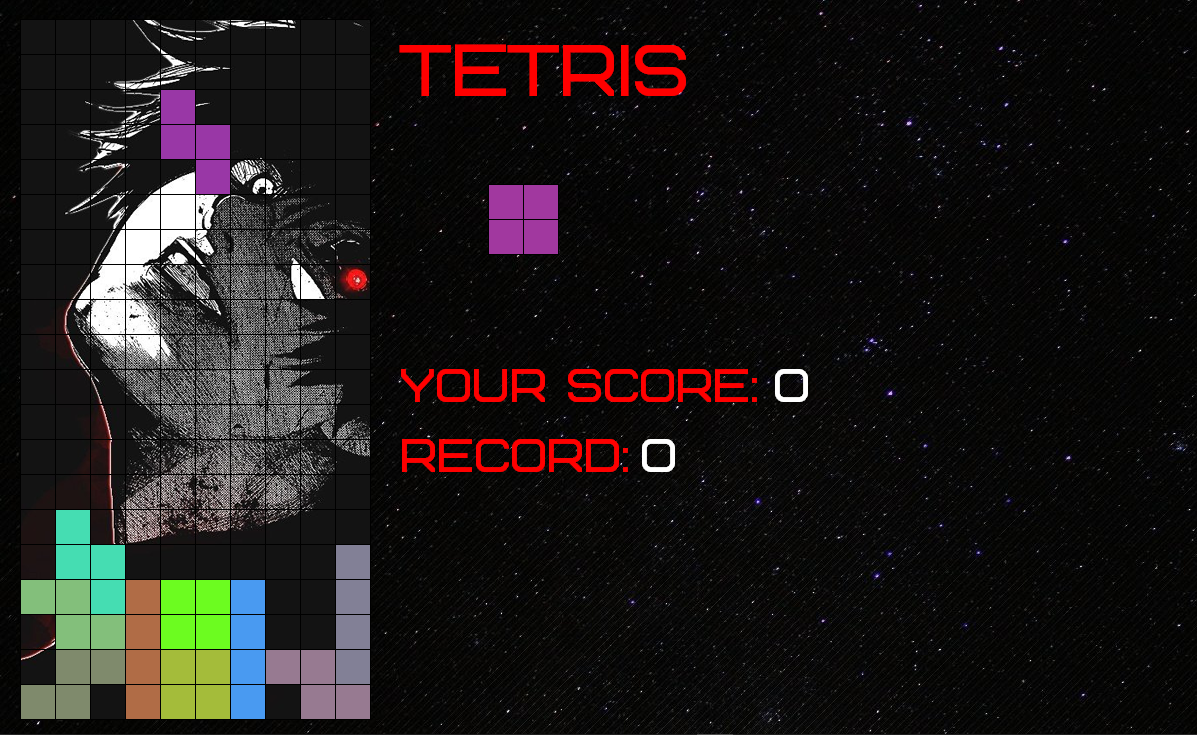
Управление проектом и соблюдение сроков: Одной из целей практики было овладение навыками управления проектом и соблюдение установленных сроков. Задачи были разбиты на подзадачи, был составлен план работ, и работа выполнялась в соответствии с этим планом. Сроки выполнения задач были соблюдены, что свидетельствует о хорошей организации работы и планирования времени.

Качество кода и тестирование: Было уделено внимание качеству кода и тестированию программного решения. Код был написан чисто и читаемо, с соблюдением стандартов программирования. Были проведены тесты на различные сценарии использования, и обнаруженные ошибки были исправлены. Это гарантирует стабильность и надежность программного решения.

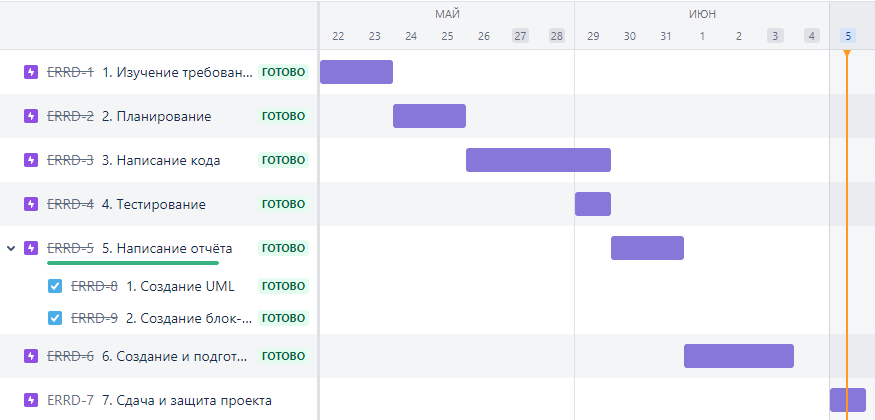
Обратная связь и самооценка: В процессе практики была осуществлена обратная связь со стороны наставника или команды разработчиков. Полученные отзывы были учтены, и внесены соответствующие изменения в работу. Также была проведена самооценка выполнения задач и достижения целей практики, что позволяет извлечь уроки и улучшить свои навыки.

Благодаря нашей учебной практике были получены не только практические навыки программирования на Python, но и применены в реальных проектах. Этот опыт будет важным в будущем и поможет успешно преуспеть в дальнейших карьерных усилиях. Наша команда смогла выполнить поставленные задачи, написать код игрового приложения, запустить игровое приложение. Также участники команды получили теоретические знания, академические умения и профессиональные навыки, изучили игровые концепции и механики, исследовали библиотеки или фреймворка для игровой разработки, программировали игровое приложение, протестировали игровое приложение, создали пользовательский интерфейс, была произведена документация процесса разработки, презентовали и демонстрировали игровое приложение, научились пользоваться такими платформами, как Jira и GitHub.

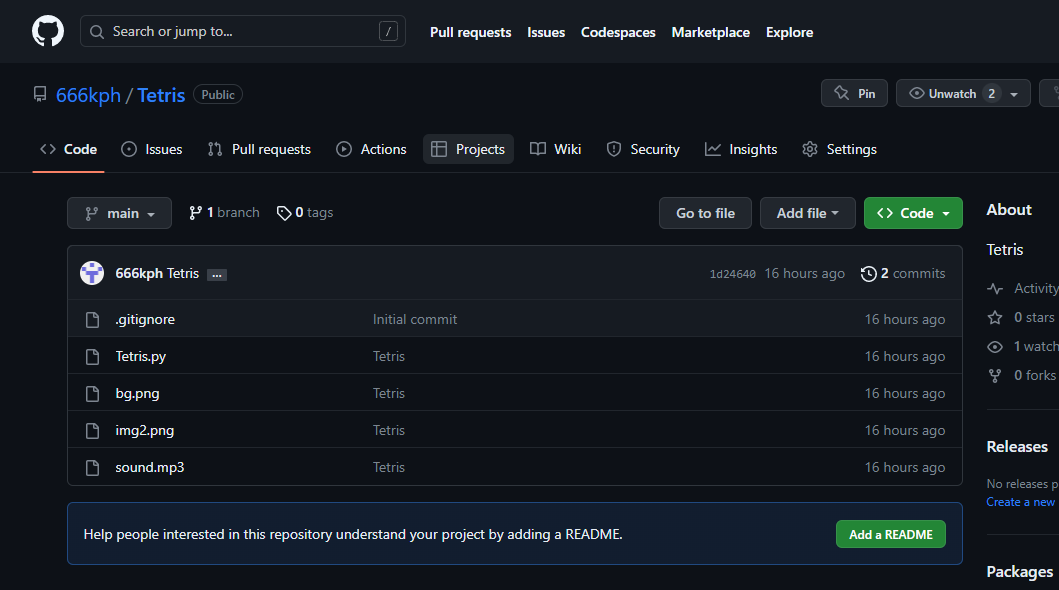
В целом, выполнение поставленных задач и достижение целей практики можно оценить как успешное. Полученные знания, навыки и опыт являются ценным ресурсом для дальнейшего развития в области разработки программного обеспечения.



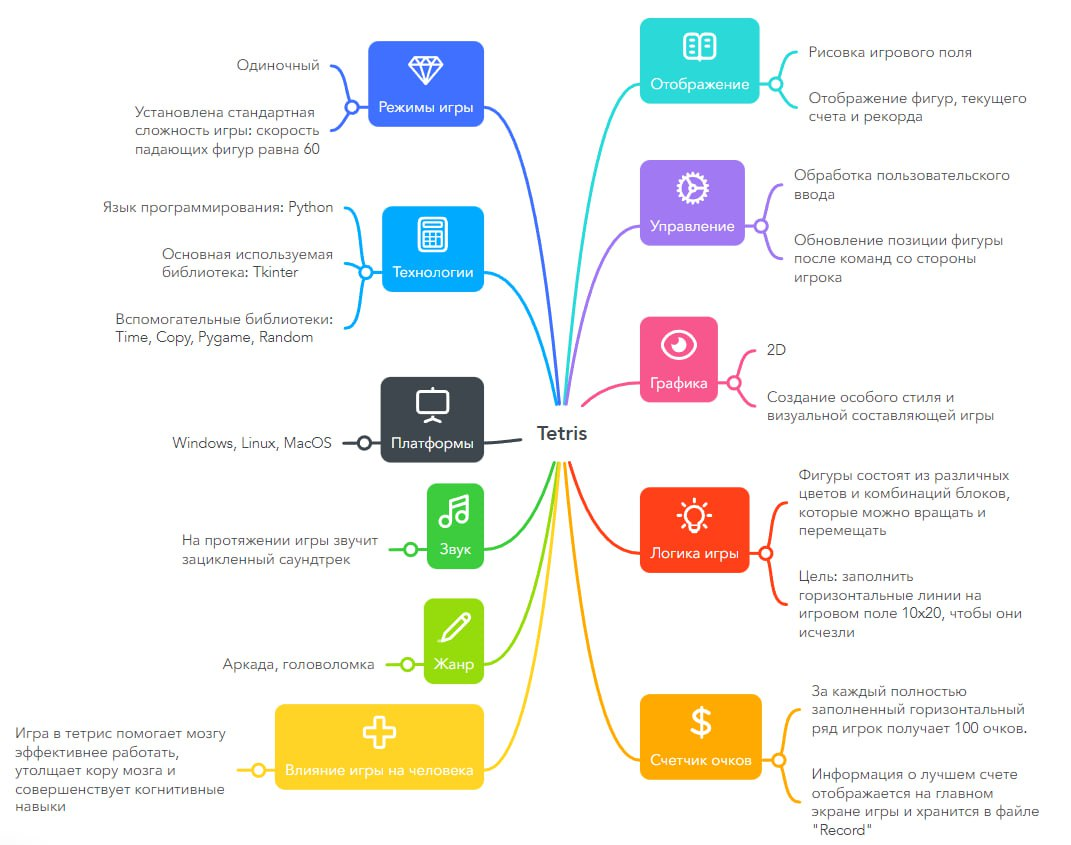
*Рисунок 18. Итоговый вид игры*



*Рисунок 19. Пример использования платформы Jira*



*Рисунок 20. Пример использования платформы GitHub*



*Рисунок 21. Интеллект-карта*

*Приложение:* https://www.mindmeister.com/ru/map/2799814827?t=3PJb92Jv0d

**6.2 Выводы**

В ходе выполнения поставленных задач и достижения целей практики можно сделать следующие выводы:

Командная работа играет ключевую роль в разработке программного обеспечения. Взаимодействие с другими участниками команды позволяет обмениваться знаниями, опытом и идеями, что способствует более эффективному решению задач и достижению целей.

Практика позволяет лучше понять реальные вызовы и проблемы, с которыми сталкиваются разработчики. Это помогает развивать аналитическое мышление и умение находить оптимальные решения, учитывая ограничения и требования проекта.

Важно быть гибким и уметь адаптироваться к изменениям. В процессе разработки могут возникать неожиданные проблемы или требования клиента могут измениться. Гибкость и способность быстро реагировать на изменения помогают успешно справиться с такими ситуациями.

Практика является важным этапом в процессе обучения и развития в области разработки программного обеспечения. Она позволяет применить теоретические знания на практике, получить опыт работы с реальными проектами и развить практические навыки.

Планирование и управление проектом играют важную роль в успешной реализации задач. Составление плана работ, распределение задач и соблюдение сроков позволяют организовать работу более эффективно и достичь поставленных целей.

# **Заключение**

Учебная практика была весьма полезным и позитивным опытом для меня. В процессе практики я смог получить ценные навыки и знания в области разработки игр и программирования на языке Python. Работа над проектом "Тетрис" позволила мне погрузиться в реальные задачи разработки игрового приложения, начиная от создания графического интерфейса до реализации игровой логики.

Одной из важных частей практики было тестирование и отладка кода, которые позволили мне улучшить навыки поиска и устранения ошибок. Возникшие проблемы научили меня быть терпеливым и находить творческие решения для их решения. Кроме того, работа в команде позволила мне развить навыки коммуникации и сотрудничества, что является важным аспектом разработки программного обеспечения.

В результате практики я приобрел уверенность в своих способностях и готовность к новым вызовам в области разработки игр. Я улучшил свои навыки программирования на языке Python, освоил библиотеку Tkinter для создания графического интерфейса и получил опыт работы с изображениями и анимацией.

В целом, учебная практика была успешной и полезной. Я благодарен за возможность принять в ней участие и за полученные знания и опыт. Этот опыт будет служить мне хорошей основой для дальнейшего развития в области разработки игр и программирования.

# **Список литературы и источники**

1. Ерогова, Инна Сергеевна. "Компьютерные игры в обучении." *Вестник науки и творчества* 9 (9) (2016): 17-22.

2. Клюшов, П. Д., and А. Е. Телепнев. "Веб-платформа для игровых и прикладных приложений." *Молодежь и современные информационные технологии: сборник трудов XVII Международной научно-практической конференции студентов, аспирантов и молодых учёных, 17-20 февраля 2020 г., г. Томск*. Томский политехнический университет, 2020.

3. Donovan, Tristan. "Replay: The History of Video Games." - 2010, East Sussex.

4. Миронов, Дмитрий Феликсович. *Компьютерная графика в дизайне [гриф есть!]*. БХВ-Петербург, 2008.

5. Sheff, David. "Game Over: How Nintendo Conquered the World." - 1993, New York.

6. Хубаев, Г. Н., and С. Н. Широбокова. "Инструментарий преобразования IDEF3-моделей бизнес-процессов в UML-диаграммы." *Глобальный научный потенциал* 2 (2015): 87.

7. Juul, Jesper. "Half-Real: Video Games between Real Rules and Fictional Worlds." - 2005, Cambridge.

8. <https://pythonru.com/uroki/obuchenie-python-gui-uroki-po-tkinter>

9. <https://python-scripts.com/tkinter>

10. <https://www.youtube.com/watch?v=qsAfMWWmAF8>

11. <https://www.youtube.com/watch?v=t2w1L2MvOGY>

12. <https://pythonru.com/osnovy/modul-time-v-python>

13. "Python Playground: Geeky Projects for the Curious Programmer" by Mahesh Venkitachalam, 2015, India.

14. "Invent Your Own Computer Games with Python" by Al Sweigart, 2010, San Francisco.

15. "Automate the Boring Stuff with Python: Practical Programming for Total Beginners" by Al Sweigart, 2015, San Francisco.