Министерство образования Республики Беларусь
Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерного проектирования

Кафедра проектирования информационных компьютерных систем

Дисциплина "Объектно-ориентированное программирование"

К защите допустить:
Руководитель курсовой работы
старший преподаватель
кафедры
_____ А.В.Михалькевич
22.01.2025

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

на тему

Разработка компьютерной игры “Tetris” на Python

БГУИР КР 1-40 05 01-10 № 155 ПЗ

Студент (подпись студента) А.Д. Устинов

Курсовая работа
представлена на проверку
22.01.2025
_____
(подпись студента)

Минск 2025

Реферат

БГУИР КР 1-40 05 01-10 № 155 ПЗ, гр. 814303

А.Д. Устинов, Разработка компьютерной игры “Tetris” на Python, Минск: БГУИР - 2025.

Пояснительная записка 38924 с., 4 рис., 0 табл.

Ключевые слова: ИГРА, ТЕТРИС, ИНТЕРФЕЙС, PYTHON

Предмет Объектно-ориентированное программирование, А.В.Михалькевич

Предмет: создание игры на Python Объект: шаблоны проектирования, разработка пользовательского интерфейса. Цель: Разработка компьютерной игры "Tetris" на Python с использованием библиотеки PyGame. Методология проведения работы: изучены принципы работы библиотеки Pygame, основные принципы игрового процесса классический игры “Tetris”, разработаны игра и её пользовательский интерфейс.. Результаты работы: разработана игра "Tetris". Интерфейс игры переработан , добавлены цвета и звуковые эффекты. Вся важная информация отображена в главном окне и в окне игры. Область применения результатов: удовлетворение пользователей, нуждающихся в новом игровом опыте, не выходя из дома, а также для развлечения. Ссылка на онлайн-репозиторий GitHub: https://github.com/radioartyomka/ZhabaTetris

Subject: Creating a Python Game Object: design patterns, user interface development. Purpose: Development of the computer game "Tetris" in Python using the PyGame library. Work methodology: the principles of the Pygame library, the basic principles of the gameplay of the classic Tetris game are studied, the game and its user interface are developed .. Results: the game "Tetris" was developed. The game interface has been redesigned, colors and sound effects added. All important information is displayed in the main window and in the game window. The scope of the results: the satisfaction of users who need a new gaming experience, without leaving home, as well as for entertainment. Link to the GitHub online repository: https://github.com/radioartyomka/ZhabaTetris

Содержание

Введение

Те́трис (производное от «тетрамино» и «теннис») — компьютерная игра, первоначально изобретённая и разработанная советским программистом Алексеем Пажитновым. Игра была выпущена 6 июня 1984 года. Первая версия была написана на языке Pascal и в ней

использовались даже не графические изображения фигур, а их текстовые аналоги, в которых квадратики были составлены из открывающей и закрывающей скобки. Сделано это было вынужденно: у компьютера «Электроника-60», на котором создавался тетрис, был дисплей, умеющий выводить только буквы и цифры и только в 24 строки по 80 символов в каждой. Сейчас классический «Тетрис» представляет собой головоломку, построенную на использовании геометрических фигур «тетрамино» — разновидности полимино, состоящих из четырёх квадратов. Игровое поле представляет собой набор квадратных ячеек. В ходе игры геометрические фигуры, выбранные из 7 предложенных вариантов, по очереди “падают” с верхней части поля игры в случайном порядке, заполняя поле. Если все ячейки на одной линии по горизонтали заполнены, эта линия очищается от фигур и становится свободной для заполнения, что немного тормозит заполнение и поля. Таким образом, основная цель игрока — наиболее эффективно совместить геометрические фигуры, чтобы быстрее заполнить линию, но не дать при этом заполнится всему игровому полю. Фигуры можно крутить на 90 градусов и перемещать по горизонтали, а также вручную ускорять их падение. С течением игры скорость “падения” фигур растёт сама, что усложняет игровой процесс и заставляет игрока принимать решения быстрее, однако повышенный стресс вознаграждается дополнительными очками за каждую заполненную линию. Такое вознаграждения за старания разжигает интерес игрока и заставляет его стремиться набрать больше очков. Кроме того, разные пользователи могут соревноваться между собой в поставленных рекордах. Такой соревновательный элемент также влияет на формирование интереса у игрока. В курсовой работе поставлена цель: разработка компьютерной игры “Тетрис”. Для достижения поставленной цели необходимо решить следующие задачи: 1 Изучение современных инструментов разработки игр. 2 Изучение принципа работы классической игры “Tetris”. 3 Разработка игры “Tetris”. 4 Разработка пользовательского интерфейса игры.

1 ОПИСАНИЕ ПРОЕКТА

Курсовой проект представляет собой переработанную версию классической игры “Tetris”. Интерфейс сделан более минималистичным и интуитивно понятным, а главное — цветным, добавлена функция задержки блока. Добавлено звуковое сопровождение.

Главная страница появляется сразу же после запуска игры и представляет собой простое меню с таблицей лидеров и единственной функцией — начать игру после нажатия клавиши “space”.

После нажатия клавиши “space” сразу же начинается сама игра, при этом её можно поставить на паузу, нажав клавишу “esc”, а при завершении игры появляется возможность ввести псевдоним из трёх букв чтоб игра запомнила результат игрока.

Игра разработана при помощи языка Python, с использованием библиотеки Pygame.

Таблица лидеров представлена в виде .txt файла, где хранятся результаты всех игроков, однако непосредственно в игре отображены только три лучших результата.

Управление и навигация довольно просты и интуитивно понятны. Приятные цвета и простой дизайн не отвлекают пользователя от игрового процесса. Напротив — многочисленные исследования показали, что игроки лучше воспринимают яркие цвета, поэтому у каждого блока свой яркий цвет, что помогает игроку лучше ориентироваться в игровом пространстве.

Сразу после запуска приложения открывается основное меню. Оно нужно для запуска самой игры после нажатия клавиши “space”, а также отображает информацию о текущем состоянии таблицы лидеров.

Рисунок 4.1 – ER – диаграмма

Рисунок 1.1 *Главное меню*

Окно игры создаётся сразу после нажатия клавиши “space” в основном меню программы. Служит собственно для игры в тетрис. Окно содержит само поле, на котором происходит игровой процесс, и базовую игровую информацию:

1. Задержанный блок. Блоки в игре можно задерживать при нажатии клавиши “left shift”. Текущий действующий блок пропадает, а на его месте появляется случайный новый, при этом задержанный блок отображается справа на экране.
2. Следующий блок. Отображает информацию о следующем блоке, который выпадет игроку, что помогает тому в нужный момент определить, как ему лучше расположить текущий действующий блок.
3. Счёт. Содержит информацию о текущем счёте игрока.
4. Уровень. Содержит информацию о текущем уровне игрока.
5. Цель. Показывает, сколько ещё линий нужно заполнить игроку чтобы перейти на следующий уровень.

Рисунок 5.1 - Обобщенный алгоритм работы.

6.

Рисунок 1.2 *Окно игры*

Окно паузы создаётся сразу после нажатия клавиши “esc” в окне игры. Окно служит для того чтобы у пользователя была возможность ненадолго приостановить игровой процесс, не сбрасывая при этом свой прогресс и не завершая игру.

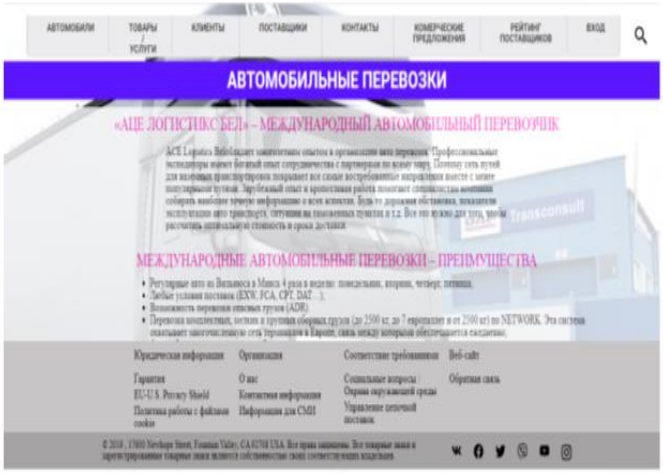

Рисунок 6.2 – Страница пользователя Администратор

Рисунок 6.3 – Страница клиенты

Рисунок 1.3 *Окно паузы*

Окно завершения игры. Создаётся сразу после проигрыша и уведомляет игрока о нём. В данном окне предоставляется возможность ввести свой псевдоним из трёх букв, чтобы игра запомнила результат игрока. Если результат оказывается одним из трёх наивысших, то он показывается в главном меню в таблице лидеров.

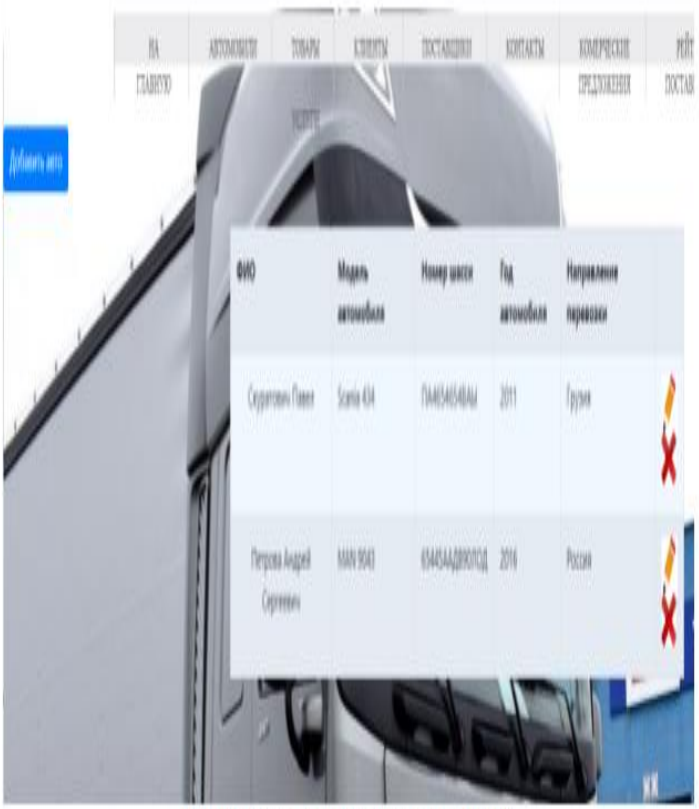

Рисунок 6.6 – Страница автомобилей

Рисунок 1.4 *Окно завершения игры*

2 ОБОСНОВАНИЕ ВЫБОРА ТЕХНОЛОГИЙ

2.1 Объектно-ориентированное программирование

Объектно-ориентированное программирование (в дальнейшем ООП) — парадигма программирования, в которой основными концепциями являются понятия объектов и классов.

В центре ООП находится понятие объекта.

Объект — это сущность, экземпляр класса, которой можно посылать сообщения и которая может на них реагировать, используя свои данные. Данные объекта скрыты от остальной программы. Сокрытие данных называется инкапсуляцией.

Наличие инкапсуляции достаточно для объектности языка программирования, но ещё не означает его объектной ориентированности — для этого требуется наличие наследования.

Но даже наличие инкапсуляции и наследования не делает язык программирования в полной мере объектным с точки зрения ООП. Основные преимущества ООП проявляются только в том случае, когда в языке программирования реализован полиморфизм, то есть возможность объектов с одинаковой спецификацией иметь различную реализацию.

Хочу выделить что очень часто натыкаюсь на мнение, что в ООП стоит выделять еще одну

немаловажную характеристику — абстракцию. Официально её не вносили в обязательные черты ООП, но списывать ее со счетов не стоит.

Абстрагирование — это способ выделить набор значимых характеристик объекта, исключая из рассмотрения не значимые. Соответственно, абстракция — это набор всех таких характеристик.

Инкапсуляция — это свойство системы, позволяющее объединить данные и методы, работающие с ними в классе, и скрыть детали реализации от пользователя.

Наследование — это свойство системы, позволяющее описать новый класс на основе уже существующего с частично или полностью заимствующейся функциональностью. Класс, от которого производится наследование, называется базовым, родительским или суперклассом. Новый класс — потомком, наследником или производным классом

Полиморфизм — это свойство системы использовать объекты с одинаковым интерфейсом без информации о типе и внутренней структуре объекта.

2.2 Обзор языка программирования Python

Python — высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. Синтаксис ядра Python минималистичен. В то же время стандартная библиотека включает большой объём полезных функций.

Python поддерживает аспектно-ориентированное, структурное, объектно-ориентированное, функциональное и императивное программирование. Основные архитектурные черты — динамическая типизация, автоматическое управление памятью, полная интроспекция, механизм обработки исключений, поддержка многопоточных вычислений, высокоуровневые структуры данных. Поддерживается разбиение программ на модули, которые, в свою очередь, могут объединяться в пакеты.

Эталонной реализацией Python является интерпретатор CPython, поддерживающий большинство активно используемых платформ. Он распространяется под свободной лицензией Python Software Foundation License, позволяющей использовать его без ограничений в любых приложениях, включая проприетарные. Есть реализация интерпретатора для JVM с возможностью компиляции, CLR, LLVM, другие независимые реализации. Проект PyPy использует JIT-компиляцию, которая значительно увеличивает скорость выполнения Python-программ.

2.3 Обзор библиотеки Pygame

Pygame – это библиотека модулей для языка Python, созданная для разработки 2D игр. Также Pygame могут называть фреймворком. В программировании понятия "библиотека" и

"фреймворк" несколько разные. Но когда дело касается классификации конкретного инструмента, не все так однозначно.

В любом случае, фреймворк является более мощным по-сравнению с библиотекой, он накладывает свою специфику на особенности программирования и сферу использования продукта. С точки зрения специфики Pygame – это фреймворк. Однако его сложно назвать "мощным инструментом". По своему объему и функционалу это скорее библиотека. [6]

3 ИНСТРУМЕНТАРИЙ

3.1 IDE PyCharm Community 2020.1.1

PyCharm — интегрированная среда разработки для языка программирования Python. Предоставляет средства для анализа кода, графический отладчик, инструмент для запуска юнит-тестов и поддерживает веб-разработку на Django. PyCharm разработана компанией JetBrains на основе IntelliJ IDEA. PyCharm — это кросс-платформенная среда разработки, которая совместима с Windows, MacOS, Linux.

Возможности PyCharm:

1. Статический анализ кода, подсветка синтаксиса и ошибок.
2. Навигация по проекту и исходному коду: отображение файловой структуры проекта, быстрый переход между файлами, классами, методами и использованиями методов.
3. Рефакторинг: переименование, извлечение метода, введение переменной, введение константы, подъём и спуск метода и т. д.
4. Инструменты для веб-разработки с использованием фреймворка Django
5. Встроенный отладчик для Python
6. Встроенные инструменты для юнит-тестирования
7. Разработка с использованием Google App Engine
8. Поддержка систем контроля версий: общий пользовательский интерфейс для Mercurial, Git, Subversion, Perforce и CVS с поддержкой списков изменений и слияния.

Пользователи могут сами писать свои плагины, тем самым расширять возможности PyCharm. Некоторые плагины из других JetBrains IDE могут работать с PyCharm. Существует более тысячи плагинов, совместимых с PyCharm.[3]

3.2 Использование системы контроля версий git

Система контроля версий(СКВ) – это система, записывающая изменения в файл или набор файлов в течение времени и позволяющая вернуться позже к определённой версии. Она позволяет вернуть файлы к состоянию, в котором они были до изменений, вернуть проект к исходному состоянию, увидеть изменения, увидеть, кто последний менял что-то и вызвал

проблему, кто поставил задачу и когда и многое другое.

Многие люди в качестве метода контроля версий применяют копирование файлов в отдельную директорию (возможно даже, директорию с отметкой по времени, если они достаточно сообразительны). Данный подход очень распространён из-за его простоты, однако он невероятно сильно подвержен появлению ошибок. Можно легко забыть, в какой директории вы находитесь, и случайно изменить не тот файл или скопировать не те файлы, которые вы хотели.

Для того, чтобы решить эту проблему, программисты разработали локальные СКВ с простой базой данных, которая хранит записи о всех изменениях в файлах, осуществляя тем самым контроль ревизий.

Одной из популярных СКВ была система RCS, которая и сегодня распространяется со многими компьютерами. RCS хранит на диске наборы патчей (различий между файлами) в специальном формате, применяя которые она может воссоздавать состояние каждого файла в заданный момент времени.

Следующая серьёзная проблема, с которой сталкиваются люди, – это необходимость взаимодействовать с другими разработчиками. Для того, чтобы разобраться с ней, были разработаны централизованные системы контроля версий (ЦСКВ). Такие системы, как CVS, Subversion и Perforce, используют единственный сервер, содержащий все версии файлов, и некоторое количество клиентов, которые получают файлы из этого централизованного хранилища. Применение ЦСКВ являлось стандартом на протяжении многих лет.

Такой подход имеет множество преимуществ, особенно перед локальными СКВ. Например, все разработчики проекта в определённой степени знают, чем занимается каждый из них. Администраторы имеют полный контроль над тем, кто и что может делать, и гораздо проще администрировать ЦСКВ, чем оперировать локальными базами данных на каждом клиенте.

Несмотря на это, данный подход тоже имеет серьёзные минусы. Самый очевидный минус – это единая точка отказа, представленная централизованным сервером. Если этот сервер выйдет из строя на час, то в течение этого времени никто не сможет использовать контроль версий для сохранения изменений, над которыми работает, а также никто не сможет обмениваться этими изменениями с другими разработчиками.

Существуют распределённые системы контроля версий (РСКВ). В РСКВ (таких как Git, Mercurial, Bazaar или Darcs) клиенты не просто скачивают снимок всех файлов (состояние файлов на определённый момент времени) – они полностью копируют репозиторий. В этом случае, если один из серверов, через который разработчики обменивались данными, умрёт, любой клиентский репозиторий может быть скопирован на другой сервер для продолжения работы. Каждая копия репозитория является полным бэкапом всех данных.

Более того, многие РСКВ могут одновременно взаимодействовать с несколькими удалёнными репозиториями, благодаря этому разработчики могут работать с различными группами людей, применяя различные подходы единовременно в рамках одного проекта. Это позволяет применять сразу несколько подходов в разработке, например, иерархические модели, что совершенно невозможно в централизованных системах.

4 АРХИТЕКТУРНЫЙ ШАБЛОН ПРОЕКТИРОВАНИЯ MVC

Model-View-Controller (MVC, «Модель-Представление-Контроллер», «Модель-Вид-Контроллер») — схема разделения данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: модель, представление и контроллер — таким образом, что модификация каждого компонента может осуществляться независимо.

Модель (Model) предоставляет данные и реагирует на команды контроллера, изменяя своё состояние.

Представление (View) отвечает за отображение данных модели пользователю, реагируя на изменения модели.

Контроллер (Controller) интерпретирует действия пользователя, оповещая модель о необходимости изменений.

Основная цель применения этой концепции состоит в отделении бизнес-логики (модели) от её визуализации (представления, вида). За счёт такого разделения повышается возможность повторного использования кода. Наиболее полезно применение данной концепции в тех случаях, когда пользователь должен видеть те же самые данные одновременно в различных контекстах и/или с различных точек зрения. В частности, выполняются следующие задачи:

1. К одной модели можно присоединить несколько видов, при этом не затрагивая реализацию модели. Например, некоторые данные могут быть одновременно представлены в виде электронной таблицы, гистограммы и круговой диаграммы;
2. Не затрагивая реализацию видов, можно изменить реакции на действия пользователя (нажатие мышью на кнопке, ввод данных) — для этого достаточно использовать другой контроллер;
3. Ряд разработчиков специализируется только в одной из областей: либо разрабатывают графический интерфейс, либо разрабатывают бизнес-логику. Поэтому возможно добиться того, что программисты, занимающиеся разработкой бизнес-логики (модели), вообще не будут осведомлены о том, какое представление будет использоваться.

5 ШАБЛОН ПРОЕКТИРОВАНИЯ ПРАКТИЧЕСКИХ ЗАДАЧ

Шаблон проектирования или паттерн (англ. *design pattern*) в разработке программного обеспечения — повторяемая архитектурная конструкция, представляющая собой решение проблемы проектирования в рамках некоторого часто возникающего контекста.

Обычно шаблон не является законченным образцом, который может быть прямо преобразован в код; это лишь пример решения задачи, который можно использовать в различных ситуациях. Объектно-ориентированные шаблоны показывают отношения и взаимодействия между классами или объектами, без определения того, какие конечные классы или объекты приложения будут использоваться.

«Низкоуровневые» шаблоны, учитывающие специфику конкретного языка программирования, называются идиомами. Это хорошие решения проектирования, характерные для конкретного языка или программной платформы, и потому не универсальные.

На наивысшем уровне существуют архитектурные шаблоны, они охватывают собой

архитектуру всей программной системы.

Алгоритмы по своей сути также являются шаблонами, но не проектирования, а вычисления, так как решают вычислительные задачи.

В сравнении с полностью самостоятельным проектированием, шаблоны обладают рядом преимуществ. Основная польза от использования шаблонов состоит в снижении сложности разработки за счёт готовых абстракций для решения целого класса проблем. Шаблон даёт решению своё имя, что облегчает коммуникацию между разработчиками, позволяя ссылаться на известные шаблоны. Таким образом, за счёт шаблонов производится унификация деталей решений: модулей, элементов проекта, — снижается количество ошибок. Применение шаблонов концептуально сродни использованию готовых библиотек кода. Правильно сформулированный шаблон проектирования позволяет, отыскав удачное решение, пользоваться им снова и снова. Набор шаблонов помогает разработчику выбрать возможный, наиболее подходящий вариант проектирования.

Шаблон проектирования или паттерн (англ. *design pattern*) в разработке программного обеспечения — повторяемая архитектурная конструкция, представляющая собой решение проблемы проектирования в рамках некоторого часто возникающего контекста.

Обычно шаблон не является законченным образцом, который может быть прямо преобразован в код; это лишь пример решения задачи, который можно использовать в различных ситуациях. Объектно-ориентированные шаблоны показывают отношения и взаимодействия между классами или объектами, без определения того, какие конечные классы или объекты приложения будут использоваться.

«Низкоуровневые» шаблоны, учитывающие специфику конкретного языка программирования, называются идиомами. Это хорошие решения проектирования, характерные для конкретного языка или программной платформы, и потому не универсальные.

На наивысшем уровне существуют архитектурные шаблоны, они охватывают собой

архитектуру всей программной системы.

Алгоритмы по своей сути также являются шаблонами, но не проектирования, а вычисления, так как решают вычислительные задачи.

В сравнении с полностью самостоятельным проектированием, шаблоны обладают рядом преимуществ. Основная польза от использования шаблонов состоит в снижении сложности разработки за счёт готовых абстракций для решения целого класса проблем. Шаблон даёт решению своё имя, что облегчает коммуникацию между разработчиками, позволяя ссылаться на известные шаблоны. Таким образом, за счёт шаблонов производится унификация деталей решений: модулей, элементов проекта, — снижается количество ошибок. Применение шаблонов концептуально сродни использованию готовых библиотек кода. Правильно сформулированный шаблон проектирования позволяет, отыскав удачное решение, пользоваться им снова и снова. Набор шаблонов помогает разработчику выбрать возможный, наиболее подходящий вариант проектирования.

6 ПРИЛОЖЕНИЕ А

Ссылка на Github: https://github.com/radioartyomka/ZhabaTetris

Заключение

В данном курсовом проекте была реализована игра “Tetris” при помощи языка программирования Python. Пользовательский интерфейс был спроектирован с использованием библиотеки Pygame. В процессе выполнения курсовой работы были изучены принципы работы библиотеки Pygame, основные принципы игрового процесса классический игры “Tetris”, были разработаны игра и её пользовательский интерфейс. В перспективе можно добавить различные модификации игры, к примеру: новые блоки и их свойства, режимы игры, возможность самостоятельно выбирать уровень сложности и скорость падения блоков. Пользовательский интерфейс также можно модифицировать добавив, например, различные варианты цветового оформления.

Список использованных источников

1. [url] **Объектно-ориентированное Программирование в Python [Электронный ресурс]. – Режим доступа:** https://python-scripts.com/object-oriented-programming-in-python
2. [url] **Git. Википедия – свободная энциклопедия [Электронный ресурс]. – Режим доступа:** https://ru.wikipedia.org/wiki/Git
3. [url] **PyCharm. Википедия – свободная энциклопедия [Электронный ресурс]. – Режим доступа:** https://ru.wikipedia.org/wiki/PyCharm
4. [url] **Статьи о различных технологиях программирования [Электронный ресурс]. – Режим доступа:** https://ru.wikipedia.org/
5. [url] **Основы Git [Электронный ресурс]. – Режим доступа:** https://git-scm.com/ [6] PyGame
6. [url] **PyGame — шпаргалка для использования [Электронный ресурс]. – Режим доступа:** https://waksoft.susu.ru/2019/04/24/pygame-shpargalka-dlja-ispolzovanija/
7. [url] **Model-View-Controller [Электронный ресурс]. – Режим доступа:** https://ru.wikipedia.org/wiki/Model-View-Controller
8. [url] **Основы паттернов проектирования Википедия – свободная энциклопедия [Электронный ресурс]. – Режим доступа:** https://ru.wikipedia.org/wiki/Шаблон_проектирования

Приложения