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 игры Арконоид на Python

БГУИР КР 1-40 05 01-10 № 128 ПЗ

Студент (подпись студента) Д.В. Залужный

Курсовая работа
представлена на проверку
22.01.2025
_____
(подпись студента)

Минск 2025

Реферат

БГУИР КР 1-40 05 01-10 № 128 ПЗ, гр. 814303

Д.В. Залужный, Разработка игры Арконоид на Python, Минск: БГУИР - 2025.

Пояснительная записка 176876 с., 13 рис., 0 табл.

Ключевые слова: Игры, Арканоид, Python, 2D, PyGame, Linux

Предмет Объектно-ориентированное программирование, А.В.Михалькевич

Предмет: создание игры Arkanoid Объект: шаблоны проектирования, разработка пользовательского интерфейса. Цель данной курсовой работы — это разработать игру Arkanoid, используя язык программирования Python.. Методология проведения работы: в процессе решения поставленных задач спроектирован и разработан простой и удобный интерфейс игры, разработана логика приложения с использованием библиотеки PyGame. Результаты работы: разработана игра с удобным функционалом. Интерфейс игры был максимально упрощен и в меру информативен, чтобы вся важная информация, была непосредственно перед пользователем. Область применения результатов: удовлетворение пользователей Интернета, нуждающихся в старых 2D играх. Ссылка на онлайн-репозиторий GitHub: https://github.com/Sleepy228/Arkanoid

-Item: creating the Arkanoid game Object: design templates, user interface development. The goal of this course work is to develop the Arkanoid game using the Python programming language.. Methodology of work: in the process of solving the tasks, a simple and convenient game interface was designed and developed, and the application logic was developed using the PyGame library. Results: developed a game with convenient functionality. The interface of the game was simplified as much as possible and moderately informative, so that all important information was directly in front of the user. Scope of the results: satisfying Internet users who need old 2D games. Link to the github online repository: https://github.com/Sleepy228/Arkanoid

Содержание

Введение

Ни для кого не секрет, что видео игры прочно заняли свою позицию в современной индустрии развлечений. Существуют попытки выделить компьютерные игры как отдельную область

искусства, наряду с театром, кино и т.п. Разработка игр может оказаться не только увлекательным, но и прибыльным делом, примеров этому предостаточно в истории. Первые примитивные компьютерные и видео игры были разработаны в 1950-х и 1960-х годах. Они работали на таких платформах, как осциллографы, университетские мейнфреймы и компьютеры EDSAC. Самой первой компьютерной игрой стал симулятор ракеты, созданный в 1942 году Томасом Голдсмитом Младшим (англ. Thomas T. Goldsmith Jr.) и Истл Рей Менном (англ. Estle Ray Mann). Позже, в 1952 году, появилась программа "OXO", имитирующая игру "крестики-нолики", созданная А.С. Дугласом как часть его докторской диссертации в Кембриджском Университете. Игра работала на большом университетском компьютере, известном как EDSAC (Electronic Delay Storage Automatic Calculator). В настоящее время, современные игры требуют достаточно большой производительности от компьютера, и не каждая офисная машина в силах воспроизводить их. Однако для отдыха от монотонной работы зачастую достаточно простой, не требовательной к технике, игры. Именно такой разработке посвящен данный курсовой проект - игра «Arkanoid». Цель данной курсовой работы — это разработать игру Arkanoid, используя язык программирования Python. Основными задачами являются создание интуитивно понятного пользовательского интерфейса, закрепление и углубление знаний и навыков, полученных при изучении дисциплины “ Объектно-ориентированного программирование”, ознакомление и изучение современных технологий, которые используются в данном проекте.

1 ОПИСАНИЕ ПРОЕКТА

Курсовой проект представляет собой игру «Арканоид» – одна из немногих игр, которая в течение долгого времени остается на пике своей популярности.

Игрок контролирует небольшую платформу, которую можно передвигать горизонтально от одной стенки до другой, подставляя её под шарик, предотвращая его падение вниз. Удар шарика по кирпичу приводит к разрушению кирпича. После того как все кирпичи на данном уровне уничтожены, происходит переход на следующий уровень, с новым набором кирпичей. Есть и некоторое разнообразие: определённые кирпичи нужно ударять несколько раз, удар по некоторым кирпичам приводит к выпаданию из них капсул-призов — приз активируется, если поймать такую капсулу платформой.

Для реализация данного проекта был выбран объектно-ориентированный язык программирования Python.

Требования:

- Установленный интерпретатор языка Python

python.org

- Установленный модуль PyGame

python3 -m pip install -U pygame --user

Особенности:

- Возможность поставить игру на паузу

- Возможность сохранения текущего состояния игры. Для сохранения нажмите Ctrl+S во время игры

- На каждом из официальных уровней присутствует музыка (по умолчанию громкость нулевая)

- Редактор уровней. Возможность создавать свои уровни

- Бонусы

Управление игрой:

A и D - управление платформой
Q – пауза
+ и - - управление громкостью музыки
Ctrl+S - сохранение игрового процесса в файл (также и на созданном уровне)

Редактор уровней:

W, A, S, D - управление курсором
1, 2, 3, 4, 5 - выбор прочности блока
E - поставить блок на место курсора

Положительные бонусы:

Увеличение длины платформы в 1.5 раз
Увеличение силы шарика

Отрицательный

- Уменьшение длины платформы в 2 раза

Запуск:

В проекте присутствует .bat-файл для запуска
Командная строка cmd: arkanoid.py

Состав:

Уровни, созданные пользователем: CreatedLevels/
Изображения: Images/
Стандартные уровни: Levels/
Музыка: Music/
Сохранения: Saves/
Файл запуска: arkanoid.py
Класс мяча: ball.py
Класс блока: block.py
Редактор уровней: editor.py
Экран ввода информации о создаваемой карте: editor_info.py
Основная логика игры: game.py
Класс карты: map.py
Меню: menu.py
Класс платформы: platform.py
Класс игрока для статистики: player.py
Экран рекордов: records_screen.py
Экран выбора созданных уровней select_custom_level.py
Экран выбора сохранения select_save.py
Послеигровой экран: statistics.py

Читы:

В игре присутствуют некоторые читы, облегчающие игровой процесс. Активируются на нажатие определенной клавиши на клавиатуре

R - респаун мяча

B - уничтожение случайного блока
N - создание щита (мяч не может провалиться вниз => игрок не проигрывает)
I - уменьшение скорости мяча до следующего соприкосновения с ракеткой (эффект slow-motion)
O - аналогично увеличение скорости мяча (если сделать скорость слишком большой, мяч начнет телепортироваться сквозь игровые объекты - использовать на свой страх и риск)

2 ОБОСНОВАНИЕ ВЫБОРА ТЕХНОЛОГИЙ

Технология программирования — это совокупность методов и средств для разработки программного обеспечения. В технологии должны быть определены последовательность выполнения операций, условия, при которых выполняется каждая операция, описание самих операций: исходные данные, нормативные документы, в том числе стандарты, критерии и методы оценки, результаты и др.

В историческом аспекте в развитии технологии программирования можно выделить несколько этапов, таких как: стихийное программирование, структурный подход к программированию, объектный подход к программированию, компонентный подход и CASE-технологии

Объектный подход к программированию сложился с середины 80-х до конца 90-х годов 20-го века. Объектно-ориентированное программирование (ООП) определяется как технология создания сложного программного обеспечения, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определенного типа (класса), а классы образуют иерархию с наследованием свойств. Взаимодействие программных объектов осуществляется путем передачи сообщений.

Основное достоинство объектно-ориентированного программирования по сравнению с модульным программированием – более естественная декомпозиция программного обеспечения, которая существенно облегчает его разработку. Кроме того, объектный подход предлагает новые способы организации программ, основанные на механизмах наследования, полиморфизма, композиции. Это позволяет существенно увеличить показатель повторного использования кодов и создавать библиотеки классов для различных применений.

Развитие объектного подхода в технологии программирования привело к созданию сред визуального программирования. Появились языки визуального объектно-ориентированного программирования, такие как Delphi, C++ Builder, Visual C++, C# и т. д. Однако технология ООП имеет и недостатки. Главный из них – зависимость модулей программного обеспечения от адресов экспортируемых полей и методов, структур и форматов данных. Эта зависимость объективна, так как модули должны взаимодействовать между собой, обращаясь к ресурсам друг друга.

Язык Python – типичный представитель ООП-семейства, обладающий элегантной и мощной объектной моделью. В этом языке от объектов никуда не спрятаться (ведь даже числа являются ими).

Выбор использования данного языка основан на том, что скорость выполнения программ, написанных на Python высока. Это связанно с тем, что основные библиотеки Python написаны

на C++ и выполнение задач занимает меньше времени, чем на других языках высокого уровня. Поэтому есть возможность писать свои собственные модули для Python на C или C++. Следующим преимуществом языка является его кроссплатформенность, т.е. скрипты, написанные при помощи Python выполняются на большинстве современных ОС. Такая переносимость обеспечивает Python применение в самых различных областях. Стоит также отметить широкое распространение пакетов (библиотек), используемых в научных вычислениях. Они способны вычислять классические численные алгоритмы решения уравнений, задач линейной алгебры, вычисления определенных интегралов, аппроксимации, решения дифференциальных уравнений. Python подходит для любых решений в области программирования, будь то офисные программы, веб-приложения, GUI-приложения и т.д.

Pygame — это «игровая библиотека», набор инструментов, помогающих программистам создавать игры. К ним относятся:

- Графика и анимация
- Звук (включая музыку)
- Управление (мышь, клавиатура, геймпад и так далее)

3 ИНСТРУМЕНТАРИЙ

1. **Обоснование используемых инструментов**

Microsoft Visual Studio — линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и вуправляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Silverlight.

Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и как отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода (как, например, Subversion и Visual SourceSafe), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования) или инструментов для прочих аспектов процесса разработки программного обеспечения (например, клиент Team Explorer для работы с Team Foundation

Server).

Notepad++ - текстовый редактор, предназначенный для программистов и всех тех, кого не устраивает скромная функциональность входящего в состав Windows Блокнота.

Основные особенности программы:

подсветка текста и возможность сворачивания блоков, согласно синтаксису языка программирования;
поддержка большого количества языков (C, C++, Java, XML, HTML, PHP, Java Script, ASCII, VB/VBS, SQL, CSS, Pascal, Perl, Python, Lua, TCL, Assembler);
WYSIWYG (печатаешь и получаешь то, что видишь на экране);
настраиваемый пользователем режим подсветки синтаксиса;
авто-завершение набираемого слова;
одновременная работа с множеством документов;
одновременный просмотр нескольких документов;
поддержка регулярных выражений Поиска/Замены;
полная поддержка перетягивания фрагментов текста;
динамическое изменение окон просмотра;
автоматическое определение состояния файла;
увеличение и уменьшение;
заметки;
выделение скобок при редактировании текста;
запись макроса и его выполнение.

Linux Mint — это одна из наиболее популярных операционных систем в мире. Она развивается сообществом и основана на операционной системе Ubuntu. Главной своей задачей разработчики Linux Mint ставят сделать максимально удобную в использовании систему и у них это получилось. Linux Mint имеет открытый исходный код и является бесплатной операционной системой, которая включает в себя все необходимые программы для повседневного использования.

PyCharm – самая популярная среда разработки, используемая для языка сценариев Python. PyCharm предлагает некоторые из лучших функций для своих пользователей и разработчиков в следующих аспектах –

Завершение кода и проверка
Расширенная отладка
Поддержка веб-программирования и фреймворков, таких как Django и Flask

1. **Использование системы контроля версий GIT**

Система контроля версий (СКВ) – это система, регистрирующая изменения в одном или нескольких файлах с тем, чтобы в дальнейшем была возможность вернуться к определённым старым версиям этих файлов [4].

СКВ даёт возможность возвращать отдельные файлы к прежнему виду, возвращать к прежнему состоянию весь проект, просматривать происходящие со временем изменения, определять, кто последним вносил изменения во внезапно переставший работать модуль, кто и

когда внёс в код какую-то ошибку, и многое другое. Вообще, если, пользуясь СКВ, испортить или потерять файлы, всё можно будет легко восстановить.

Многие предпочитают контролировать версии, просто копируя файлы в другой каталог (как правило добавляя текущую дату к названию каталога). Такой подход очень распространён, потому что прост, но он и чаще даёт сбои. Очень легко забыть, что ты не в том каталоге, и случайно изменить не тот файл, либо скопировать файлы не туда, куда хотел, и затереть нужные файлы.

Чтобы решить эту проблему, программисты уже давно разработали локальные СКВ с простой базой данных, в которой хранятся все изменения нужных файлов (рисунок 1).

Рисунок 1- Схема локальной СКВ

Одной из наиболее популярных СКВ такого типа является rcs, которая до сих пор устанавливается на многие компьютеры. Даже в современной операционной системе Mac OS X утилита rcs устанавливается вместе с Developer Tools. Эта утилита основана на работе с наборами патчей между парами версий (патч – файл, описывающий различие между файлами), которые хранятся в специальном формате на диске. Это позволяет пересоздать любой файл на любой момент времени, последовательно накладывая патчи.

Следующей основной проблемой оказалась необходимость сотрудничать с разработчиками за другими компьютерами. Чтобы решить её, были созданы централизованные системы контроля версий (ЦСКВ). В таких системах, например CVS, Subversion и Perforce, есть центральный сервер, на котором хранятся все файлы под версионным контролем, и ряд клиентов, которые получают копии файлов из него. Много лет это было стандартом для систем контроля версий (рис. 2).

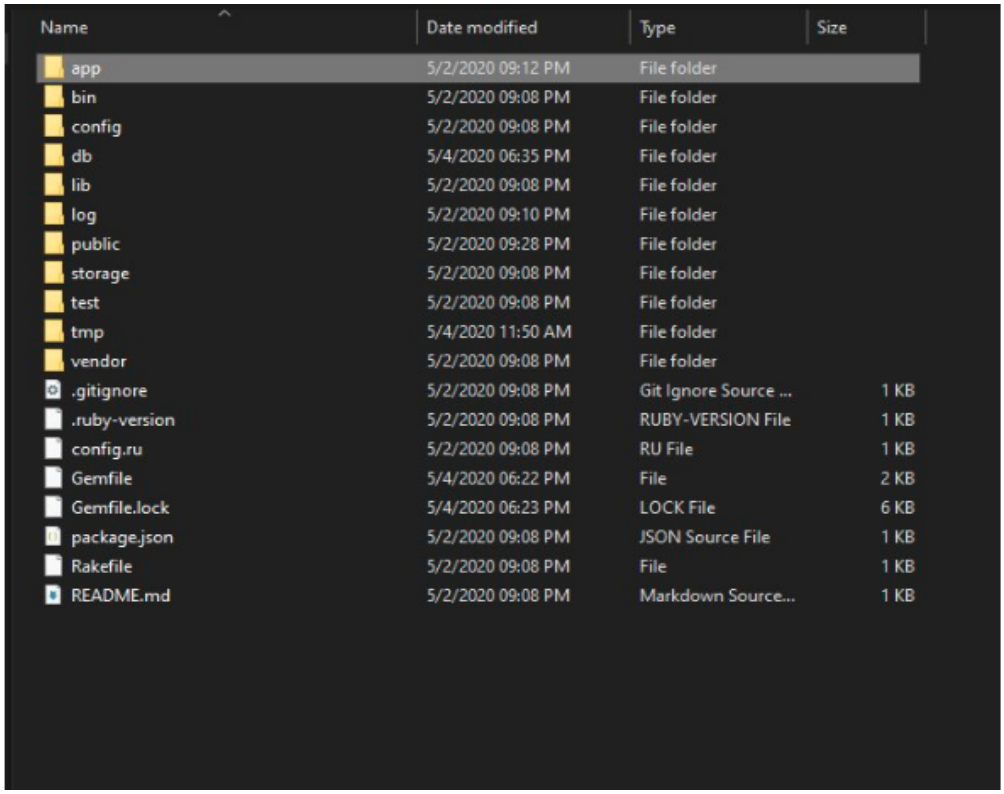

Рисунок 2 - Схема централизованного контроля версий

Такой подход имеет множество преимуществ, особенно над локальными СКВ. К примеру, все знают, кто и чем занимается в проекте. У администраторов есть чёткий контроль над тем, кто и что может делать, и, конечно, администрировать ЦСКВ намного легче, чем локальные базы на каждом клиенте.

Однако при таком подходе есть и несколько серьёзных недостатков. Наиболее очевидный – централизованный сервер является уязвимым местом всей системы. Если сервер выключается на час, то в течение часа разработчики не могут взаимодействовать, и никто не может сохранить новой версии своей работы. Если же повреждается диск с центральной базой данных и нет резервной копии, вы теряете абсолютно всё – всю историю проекта, разве что за исключением нескольких рабочих версий, сохранившихся на рабочих машинах пользователей. Локальные системы контроля версий подвержены той же проблеме: если вся история проекта хранится в одном месте, вы рискуете потерять всё.

И в этой ситуации в игру вступают распределённые системы контроля версий (РСКВ). В таких системах как Git, Mercurial, Bazaar или Darcs клиенты не просто выгружают последние версии файлов, а полностью копируют весь репозиторий. Поэтому в случае, когда "умирает" сервер, через который шла работа, любой клиентский репозиторий может быть скопирован обратно на сервер, чтобы восстановить базу данных. Каждый раз, когда клиент забирает свежую версию файлов, он создаёт себе полную копию всех данных (рисунок 3).

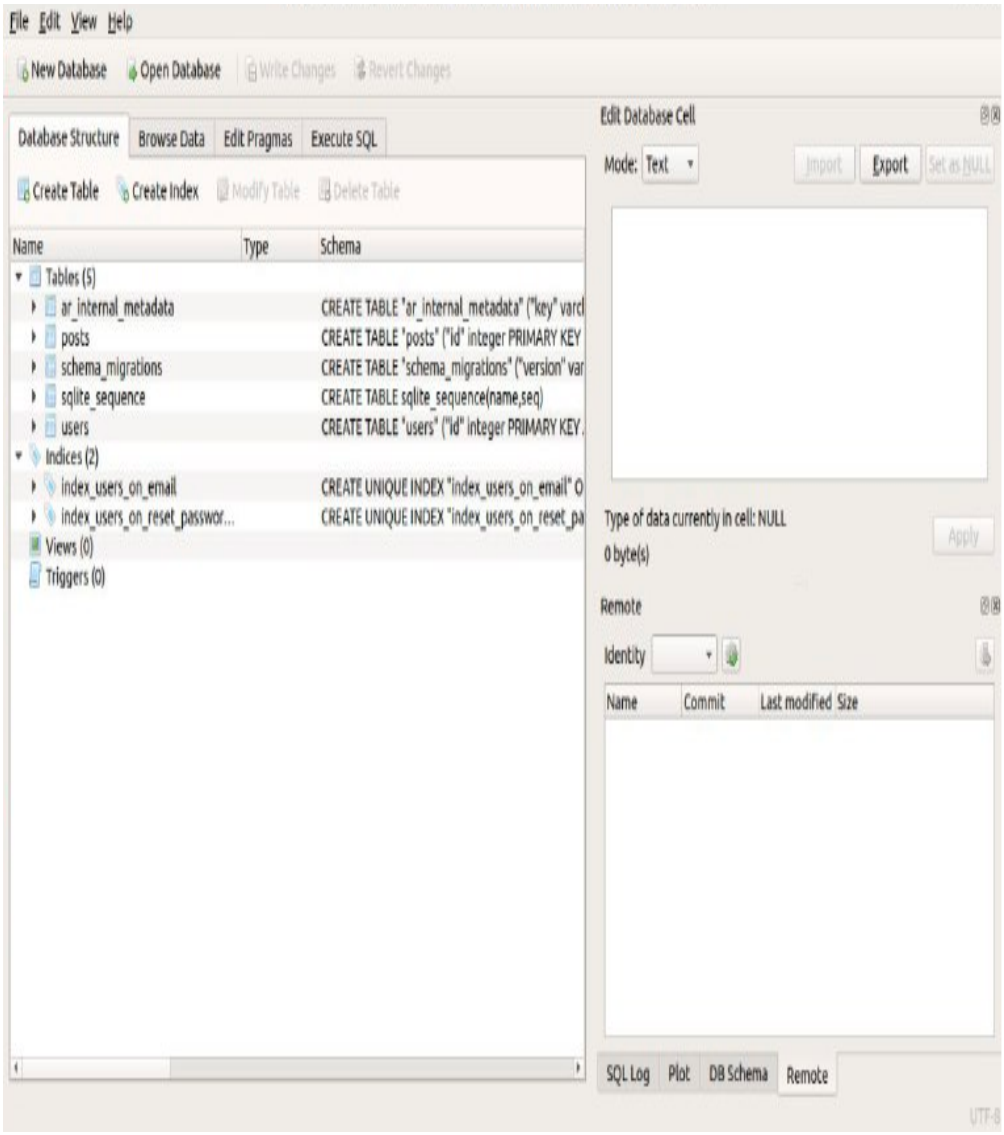

Рисунок 3 – Схема распределённой системы контроля версий

Кроме того, в большей части этих систем можно работать с несколькими удалёнными репозиториями, таким образом, можно одновременно работать по-разному с разными группами людей в рамках одного проекта. Так, в одном проекте можно одновременно вести несколько типов рабочих процессов, что невозможно в централизованных системах.

4 АРХИТЕКТУРНЫЙ ШАБЛОН ПРОЕКТИРОВАНИЯ MVC

Шаблон проектирования Модель – Представление – Контроллер (MVC) – это шаблон программной архитектуры, построенный на основе сохранения представления данных отдельно от методов, которые взаимодействуют с данными .

Не смотря на то, что схема MVC была первоначально разработана для персональных компьютеров, она была адаптирована и широко используется веб-разработчиками из-за

точного разграничения задач и возможности повторного использования кода. Схема стимулирует развитие модульных систем, что позволяет разработчикам быстро обновлять, добавлять или удалять функционал.

Название шаблона проектирования определяется тремя его основными составляющими частями: Модель, Представление и Контроллер. Визуальное представление шаблона MVC выглядит, как показано на приведенной ниже диаграмме (рис.4):

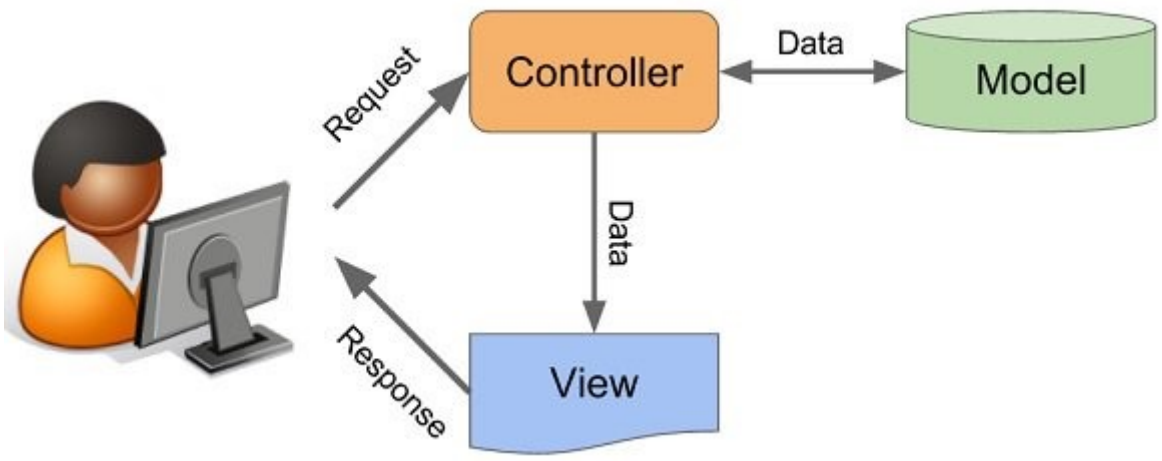

Рисунок 4 - Схема шаблона MVC

На рисунке показана структура одностороннего потока данных и пути его следования между различными компонентами, а также их взаимодействие.

Моделью называют постоянное хранилище данных, используемых во всей структуре. Она должна обеспечивать доступ к данным для их просмотра, отбора или записи. В общей структуре Модель является мостом между компонентами Представление и Контроллер.

При этом Модель не имеет никакой связи или информации о том, что происходит с данными, когда они передаются компонентам Представление или Контроллер. Единственная задача Модели – обработка данных в постоянном хранилище, поиск и подготовка данных, передаваемых другим составляющим MVC.

Модель должна выступать в качестве «привратника», стоящего возле хранилища данных и не задающего вопросов, но принимающего все поступающие запросы. Зачастую это наиболее сложная часть системы MVC. Компонент Модель – это вершина всей структуры, так как без нее невозможна связь между Контроллером и Представлением.

Представление – это часть системы, в которой данным, запрашиваемым у Модели, задается окончательный вид их вывода. В веб-приложениях, созданных на основе MVC, Представление – это компонент, в котором генерируется и отображается HTML-код.

Представление также перехватывает действие пользователя, которое затем передается Контроллеру. Характерным примером этого является кнопка, генерируемая Представлением. Когда пользователь нажимает ее, запускается действие в Контроллере.

Существует несколько распространенных заблуждений относительно компонента Представление. Например, многие ошибочно полагают, что Представление не имеет никакой

связи с Моделью, а все отображаемые данные передаются от Контроллера. В действительности такая схема потока данных не учитывает теорию, лежащую в основе MVC архитектуры.

Кроме этого определение Представления как файла шаблона также является неточным. Но это не вина одного человека, а результат множества ошибок различных разработчиков, которые приводят общему заблуждению. После чего они неправильно объясняют это другим. На самом деле Представление это намного больше, чем просто шаблон. Но современные MVC-ориентированные фреймворки до такой степени впитали этот подход, что никто уже не заботится о том, поддерживается ли верная структура MVC или нет.

Компоненту Представление никогда не передаются данные непосредственно Контроллером. Между Представлением и Контроллером нет прямой связи – они соединяются с помощью Модели.

Его задача заключается в обработке данных, которые пользователь вводит и обновлении Модели. Это единственная часть схемы, для которой необходимо взаимодействие пользователя.

Контроллер можно определить, как сборщик информации, которая затем передается в Модель с последующей организацией для хранения. Он не содержит никакой другой логики, кроме необходимости собрать входящие данные. Контроллер также подключается только к одному Представлению и одной Модели. Это создает систему с односторонним потоком данных с одним входом и одним выходом в точках обмена данными.

Контроллер получает задачи на выполнение только когда пользователь взаимодействует с Представлением, и каждая функция зависит от взаимодействия пользователя с Представлением. Наиболее распространенная ошибка разработчиков заключается в том, что они путают Контроллер со шлюзом, поэтому присваивают ему функции и задачи, которые относятся к Представлению.

Также распространенной ошибкой является наделение Контроллера функциями, которые отвечают только за обработку и передачу данных из Модели в Представление. Но согласно структуре MVC паттерна это взаимодействие должно осуществляться между Моделью и Представлением.

5 ШАБЛОНЫ ПРОЕКТИРОВАНИЯ

Шаблон проектирования или паттерн 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 архитектуру всей программной системы.

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.

Хотя легкое изменение кода под известный шаблон может упростить понимание кода, есть мнение, что с применением шаблонов могут быть связаны две сложности. Во-первых, слепое следование некоторому выбранному шаблону может привести к усложнению программы. Во-вторых, у разработчика может возникнуть желание попробовать некоторый шаблон в деле без особых оснований.

Поэтому шаблоны проектирования стоит использовать осмысленно и только там, где они требуются.

Шаблоны бывают следующих видов:

Порождающие;
Структурные;
Поведенческие;

Порождающие шаблоны — шаблоны проектирования, которые абстрагируют процесс инстанцирования. Они позволяют сделать систему независимой от способа создания, композиции и представления объектов. Шаблон, порождающий классы, использует наследование, чтобы изменять наследуемый класс, а шаблон, порождающий объекты, делегирует инстанцирование другому объекту.

Существуют следующие порождающие шаблоны:

Простая фабрика (Simple Factory);
Фабричный метод (Factory Method);
Абстрактная фабрика (Abstract Factory);
Строитель (Builder);
Прототип (Prototype);
Одиночка (Singleton).

Структурные шаблоны — шаблоны проектирования, в которых рассматривается вопрос о том, как из классов и объектов образуются более крупные структуры.

Список структурных шаблонов проектирования:

Адаптер (Adapter);
Мост (Bridge);
Компоновщик (Composite);
Декоратор (Decorator);
Фасад (Facade);
Приспособленец (Flyweight);
Заместитель (Proxy).

Поведенческие шаблоны — шаблоны проектирования, определяющие алгоритмы и способы реализации взаимодействия различных объектов и классов.

Поведенческие шаблоны:

Цепочка обязанностей (Chain of Responsibility);
Команда (Command);
Итератор (Iterator);
Посредник (Mediator);
Хранитель (Memento);
Наблюдатель (Observer);
Посетитель (Visitor);
Стратегия (Strategy);
Состояние (State);
Шаблонный метод (Template Method).

6 ОПИСАНИЕ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА

При загрузке игры, пользователь попадает в главное меню, где отображается основной функционал. (рис.5)

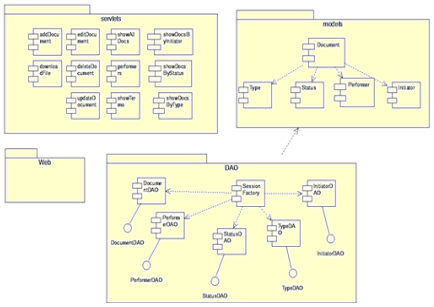

Рисунок 5 – Главное меню

Для запуска новой игры, необходимо нажать на клавишу “1”. После обрабатывания нажатия, произойдет загрузка первого уровня, где можно будет наблюдать объекты, которые будут присутствовать на протяжении всего игрового процесса: основное игровое поле, блоки, платформу, шар и основную информацию, которая отображает пункты игрового процесса. (рис.6)

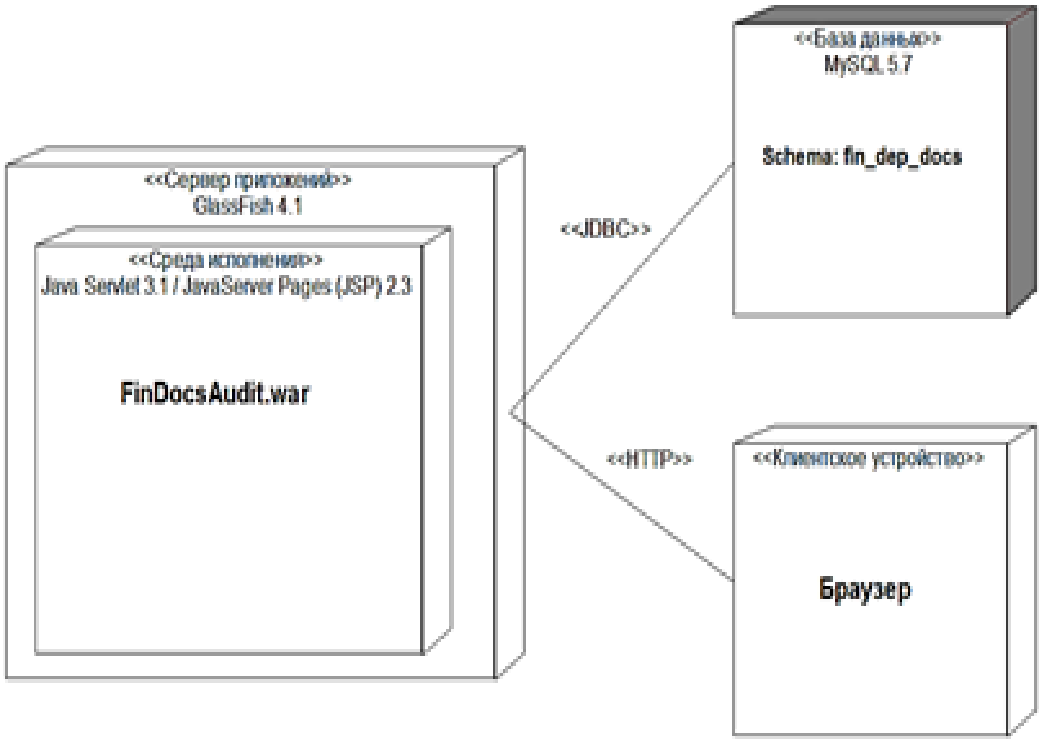

Рисунок 6 – Начало игры

Так же был реализован дополнительный функционал: режим “Пауза”, режим сохранения, прибавление и убавления уровня громкости музыки в игре, читерские клавиши. Демонстрация режима паузы можно наблюдать на Рисунке 7.

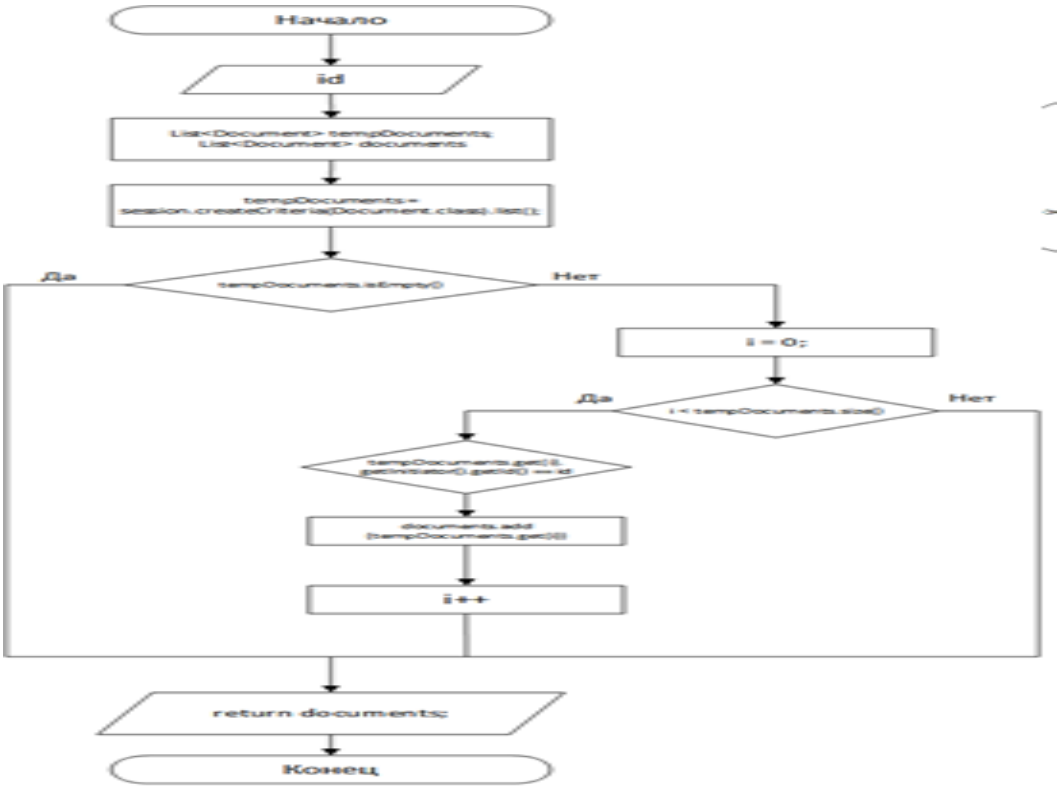

Рисунок 7 – Режим “Пауза”

После того, как пользователь разбил все блоки на уровне, он переходит на новый уровень, но если были потрачены все жизни, то пользователю будет предложено ввести свой никнейм для попадания в статистику. (рис.8)

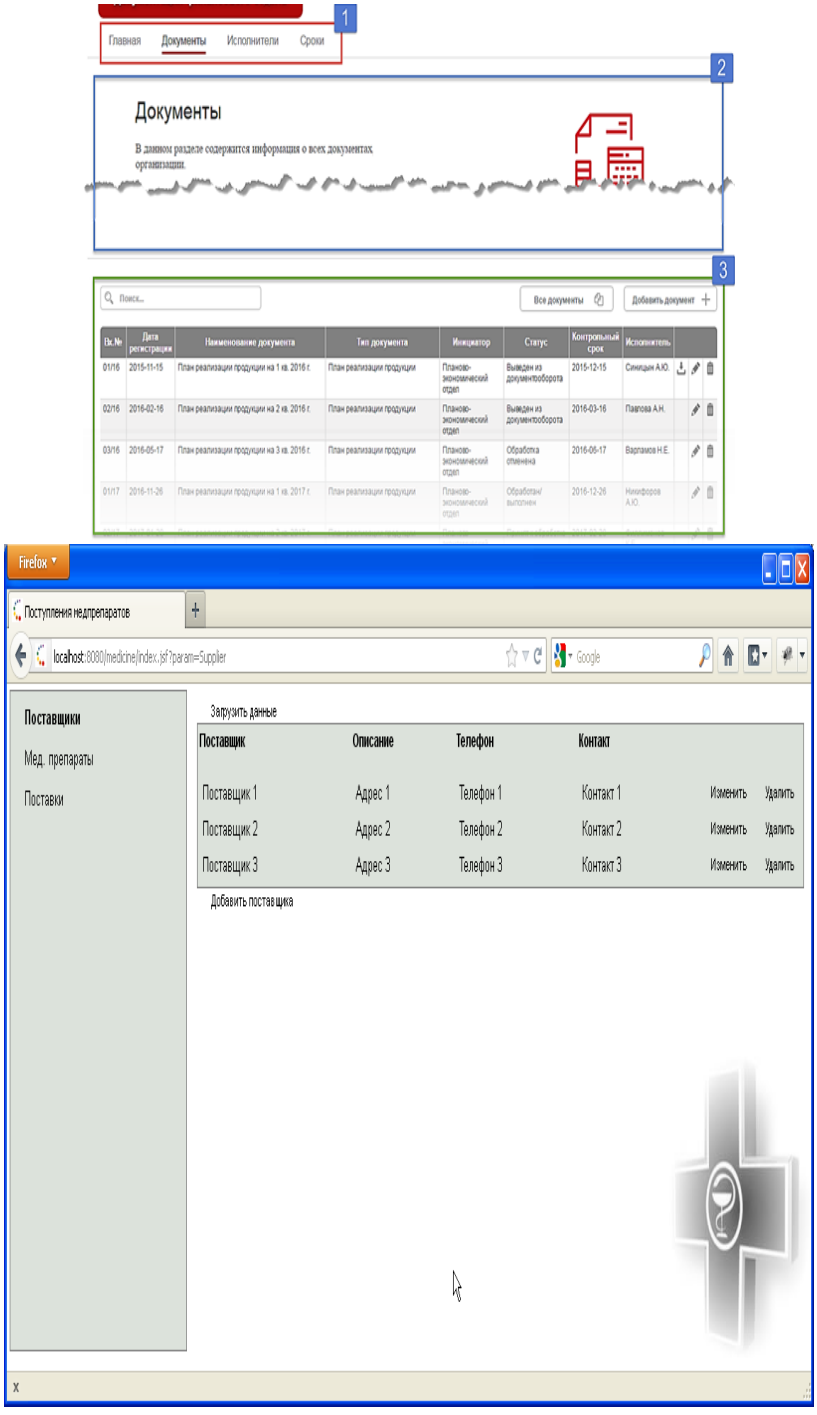

Рисунок 8 – Статистика

В главном меню пользователь может загрузить свое сохранение, при нажатии на клавишу “2”. После того, как будет загружено меню, пользователь может выбрать из списка всех сохранений то, которое ему необходимо. Файл, который будет выбран, подсвечивается зеленым цветом (рис.9)

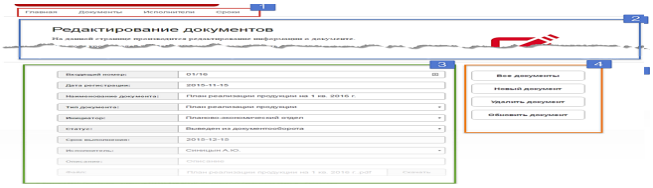

Рисунок 9 – Загрузка сохранения

Была предусмотрена индивидуальная разработка уровня. Для этого пользователю необходимо в основном меню нажать на клавишу “3”. В новой вкладке выбрать размеры поля, при указании недопустимого размера, клетка подсвечивается красным цветом. Далее пользователю необходимо разместить блоки на поле по своему усмотрению и нажать на сочетание клавиш “Ctrl” + ”S” для сохранения уровня. (рис.10)

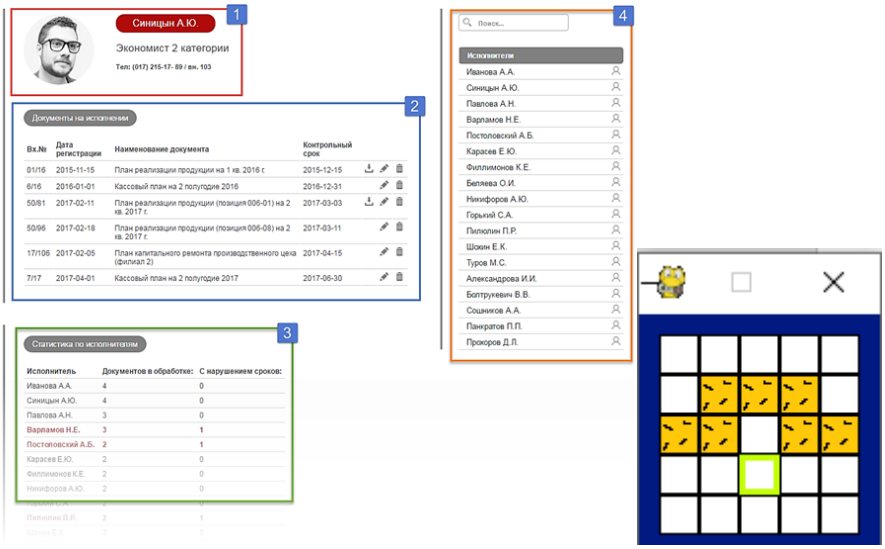

Рисунок 10 – Создание уровня

Последняя вкладка, которая доступна пользователю – это загрузка уровней. Меню загрузки уровня реализовано так же, как и меню загрузки сохранения. (рис.11)

Рисунок 11 – Загрузка уровня

7 ПРИЛОЖЕНИЕ А

Листинг кода

```
import menu


if __name__ == "__main__":
    m = menu.Menu()
import random


class Ball:

    def __init__(self, screen_width, screen_height, x=-1, y=-1, sp0=-1,
                 sp1=-1, start=-1, power=1):
        self.screen_width = screen_width
        if x == -1:
            self.x = random.randint(40, screen_width - 40)
            self.start_y = self.y = screen_height - 60
            self.speed = [1, -1]
            self.power = 1
        else:
            self.x = x
            self.y = y
            self.speed = [sp0, sp1]
            self.start_y = start
            self.power = power
        self.basic_speed = 1
        self.top = self.y - 10
        self.bottom = self.y + 10
        self.left = self.x - 10
        self.right = self.x + 10
        self.color = "#c8ff00"

    def get_side_of_intersection(self, obj):
        if self.top == obj.bottom:
            return "top"
        if self.bottom == obj.top:
            return "bottom"
        if self.left == obj.right:
            return "left"
        if self.right == obj.left:
            return "right"
```

```
    # def __init__(self, x, y, speed0, speed1, start):
    #     self.x = x
    #     self.y = y
    #     self.start_y = start
    #     self.top = self.y - 10
    #     self.bottom = self.y + 10
    #     self.left = self.x - 10
    #     self.right = self.x + 10
    #     self.basic_speed = 1
    #     self.speed = [speed0, speed1]
    #     self.color = "#c8ff00"

    def move(self):
        self.x += self.speed[0]
        self.y += self.speed[1]
        self.recount_coordinates()

    def reincarnate(self):
        self.x = random.randint(20, self.screen_width - 20)
        self.y = self.start_y
        self.recount_coordinates()
        self.basic_speed = 1
        self.speed = [1, -1]

    def recount_coordinates(self):
        self.top = self.y - 10
        self.bottom = self.y + 10
        self.left = self.x - 10
        self.right = self.x + 10

import pygame as pg
import random


class Block:

    bonuses = [
        "powerup",
        "platform_more",
        "platform_less",
        "destroy_line"
```

```
]

def __init__(self, x, y, str, bonus=None):
    self.x = x
    self.y = y
    if bonus is None:
        chance = random.randint(0, 14)
        if chance == 10:
            self.bonus = \
                self.bonuses[random.randint(0, len(self.bonuses) - 1)]
        else:
            self.bonus = None
    else:
        self.bonus = bonus
    self.top = self.y - 10
    self.bottom = self.y + 10
    self.left = self.x - 10
    self.right = self.x + 10
    self.strength = str
    file_name = "Images/block{0}.png".format(self.strength)
    self.image = pg.image.load(file_name)

def recount_coordinates(self):
    self.top = self.y - 10
    self.bottom = self.y + 10
    self.left = self.x - 10
    self.right = self.x + 10

def decrease_and_check_destroying(self, power):
    self.strength -= power
    if self.strength <= 0:
        return True
    else:
        file_name = "Images/block{0}.png".format(self.strength)
        self.image = pg.image.load(file_name)
        return False

def draw(self, screen):
    screen.blit(self.image, (self.left, self.top))

def __str__(self):
```

```
        return str(self.x) + " " + str(self.y)

import pygame


class Bonus:

    images = {
        "powerup": "Images/powerup.png",
        "platform_more": "Images/platform_more.png",
        "platform_less": "Images/platform_less.png"
    }

    def __init__(self, name, x, y):
        self.name = name
        self.image = pygame.image.load(self.images[name])
        self.x = x
        self.y = y
        self.speed = [0, 1]

import pygame
import sys
import random
import menu


class Editor:
    images = {0:  pygame.image.load("Images/block.png"),
              1:  pygame.image.load("Images/block1.png"),
              2:  pygame.image.load("Images/block2.png"),
              3:  pygame.image.load("Images/block3.png"),
              4:  pygame.image.load("Images/block4.png"),
              5:  pygame.image.load("Images/block5.png"),
              10: pygame.image.load("Images/cursor.png")}
    alph = "abcdefghijklmnopqrstuvwxyz"

    def __init__(self, width, height):
        self.width = width
        self.height = height
        print(width, height)
        self.field_width = width * 20 + 20
```

```
    self.field_height = height * 20 + 20
    print(self.field_width, self.field_height)
    self.game_objects = []
    for i in range(0, self.height):
        line = []
        for j in range(0, self.width):
            line.append(' ')
        self.game_objects.append(line)
    print(len(self.game_objects), len(self.game_objects[0]))
    self.cursor = [0, 0]
    self.selected = "1"
    self.display = (self.field_width, self.field_height)
    self.background_color = "#001a82"
    self.screen = pygame.display.set_mode(self.display)
    self.bg = pygame.Surface(self.display)
    self.timer = pygame.time.Clock()
    self.ctrl_pressed = False
    self.start()

def start(self):
    pygame.init()
    pygame.display.set_caption("Editor")
    self.bg.fill(pygame.Color("#14005e"))
    while True:
        self.timer.tick(200)
        for e in pygame.event.get():
            self.handle_keys(e)
        self.draw()

def handle_keys(self, e):
    if e.type == pygame.QUIT:
        sys.exit()
    if e.type == pygame.KEYDOWN and e.key == pygame.K_ESCAPE:
        menu.Menu()
    elif e.type == pygame.KEYDOWN and e.key == pygame.K_a:
        if 0 < self.cursor[1]:
            self.cursor[1] -= 1
            print(self.cursor)
    elif e.type == pygame.KEYDOWN and e.key == pygame.K_d:
        if self.cursor[1] < self.width - 1:
            self.cursor[1] += 1
```

```
            print(self.cursor)
        elif e.type == pygame.KEYDOWN and e.key == pygame.K_w:
            if 0 < self.cursor[0]:
                self.cursor[0] -= 1
                print(self.cursor)
        elif e.type == pygame.KEYDOWN and e.key == pygame.K_s:
            if self.cursor[0] < self.height - 1:
                self.cursor[0] += 1
                print(self.cursor)
        elif e.type == pygame.KEYDOWN and e.key == pygame.K_e:
            x = self.cursor[0]
            y = self.cursor[1]
            if self.selected == '0':
                self.game_objects[x][y] = ' '
            else:
                self.game_objects[x][y] = self.selected
        elif e.type == pygame.KEYDOWN and e.key == pygame.K_LCTRL:
            self.ctrl_pressed = True
        elif e.type == pygame.KEYUP and e.key == pygame.K_LCTRL:
            self.ctrl_pressed = False
        if e.type == pygame.KEYDOWN and e.key == pygame.K_s:
            if self.ctrl_pressed:
                self.save_map()
        elif e.type == pygame.KEYDOWN and e.key == pygame.K_1:
            self.selected = "1"
        elif e.type == pygame.KEYDOWN and e.key == pygame.K_2:
            self.selected = "2"
        elif e.type == pygame.KEYDOWN and e.key == pygame.K_3:
            self.selected = "3"
        elif e.type == pygame.KEYDOWN and e.key == pygame.K_4:
            self.selected = "4"
        elif e.type == pygame.KEYDOWN and e.key == pygame.K_5:
            self.selected = "5"
        elif e.type == pygame.KEYDOWN and e.key == pygame.K_0:
            self.selected = "0"

    def save_map(self):
        l = random.randint(6, 12)
        name = ""
        for i in range(0, l):
            name += self.alph[random.randint(0, len(self.alph) - 1)]
```

```
        map = ""
        for i in range(0, self.width + 2):
            map += "B"
        for i in range(0, len(self.game_objects)):
            map += "\nB"
            for j in range(0, len(self.game_objects[i])):
                map += self.game_objects[i][j]
            map += "B"
        print(map)
        for i in range(0, 8):
            map += "\nB"
            for j in range(0, self.width):
                map += " "
            map += "B"
        print(map)
        file = open("CreatedLevels/{0}.txt".format(name), 'w')
        file.write(map)
        file.close()
        menu.Menu()

    def draw(self):
        self.screen.blit(self.bg, (0, 0))
        self.bg.fill(pygame.Color(self.background_color))
        x = 10
        y = 10
        for i in range(0, self.height):
            for j in range(0, self.width):
                block = self.game_objects[i][j]
                if block == " ":
                    self.screen.blit(self.images[0], (x, y))
                else:
                    self.screen.blit(self.images[int(block)], (x, y))
                x += 20
            y += 20
            x = 10
        x = self.cursor[1]
        y = self.cursor[0]
        if self.game_objects[y][x] == ' ':
            self.screen.blit(self.images[10], (x*20 + 10, y*20 + 10))
        pygame.display.update()
```

```
import pygame
import sys
import editor
import menu


class EditorMapInfo:
    def __init__(self):
        self.display = (320, 240)
        self.screen = pygame.display.set_mode(self.display)
        self.bg = pygame.Surface(self.display)
        self.timer = pygame.time.Clock()
        self.active = 1
        self.width = 0
        self.width_text = "5"
        self.height = 0
        self.height_text = "5"
        self.active_color = "#325200"
        self.inactive_color = "#383838"
        self.start()

    def start(self):
        pygame.init()
        pygame.display.set_caption("New map info")
        self.bg.fill(pygame.Color("#14005e"))
        while True:
            self.timer.tick(200)
            for e in pygame.event.get():
                if e.type == pygame.QUIT:
                    sys.exit()
                if e.type == pygame.KEYDOWN and e.key == pygame.K_ESCAPE:
                    menu.Menu()
                if e.type == pygame.KEYDOWN and e.key == pygame.K_LEFT:
                    self.active -= 1
                if e.type == pygame.KEYDOWN and e.key == pygame.K_RIGHT:
                    self.active += 1
                if e.type == pygame.KEYDOWN and e.key == pygame.K_RETURN:
                    self.width = int(self.width_text)
                    self.height = int(self.height_text)
                    if 5 <= self.width <= 25 and 5 <= self.height <= 25:
                        ed = editor.Editor(self.width, self.height)
```

```
    elif e.type == pygame.KEYDOWN:
        if e.key == pygame.K_BACKSPACE:
            if self.active % 2 == 1:
                self.width_text = self.width_text[:-1]
            else:
                self.height_text = self.height_text[:-1]
        else:
            if self.active % 2 == 1:
                self.width_text += e.unicode
            else:
                self.height_text += e.unicode
self.screen.blit(self.bg, (0, 0))
font = pygame.font.Font(None, 25)
text = font.render("Input count of blocks in field",
                   True, (255, 255, 255))
self.screen.blit(text, [10, 15])
text = font.render("min = 5, max = 25", True, (255, 255, 255))
self.screen.blit(text, [10, 40])
text = font.render("Width", True, (255, 255, 255))
self.screen.blit(text, [10, 70])
text = font.render("Height", True, (255, 255, 255))
self.screen.blit(text, [100, 70])
if self.active % 2 == 1:
    font = pygame.font.Font(None, 40)

    pf = pygame.Surface((80, 50))
    pf.fill(pygame.Color(self.active_color))
    self.screen.blit(pf, (10, 100))

    text = font.render(self.width_text, True, (255, 255, 255))
    self.screen.blit(text, [30, 110])

    pf.fill(pygame.Color(self.inactive_color))
    self.screen.blit(pf, (100, 100))

    text = font.render(self.height_text, True, (255, 255, 255))
    self.screen.blit(text, [120, 110])
else:
    font = pygame.font.Font(None, 40)

    pf = pygame.Surface((80, 50))
```

```
    pf.fill(pygame.Color(self.inactive_color))
    self.screen.blit(pf, (10, 100))

    text = font.render(self.width_text, True, (255, 255, 255))
    self.screen.blit(text, [30, 110])

    pf.fill(pygame.Color(self.active_color))
    self.screen.blit(pf, (100, 100))

    text = font.render(self.height_text, True, (255, 255, 255))
    self.screen.blit(text, [120, 110])

try:
    self.width = int(self.width_text)
    if not 5 <= self.width <= 25:
        pf = pygame.Surface((80, 50))
        pf.fill(pygame.Color("#ff0000"))
        self.screen.blit(pf, (10, 100))
        text = font.render(self.width_text, True, (255, 255, 255))
        self.screen.blit(text, [30, 110])
except:
    pf = pygame.Surface((80, 50))
    pf.fill(pygame.Color("#ff0000"))
    self.screen.blit(pf, (10, 100))
    text = font.render(self.width_text, True, (255, 255, 255))
    self.screen.blit(text, [30, 110])

try:
    self.height = int(self.height_text)
    if not 5 <= self.height <= 25:
        pf = pygame.Surface((80, 50))
        pf.fill(pygame.Color("#ff0000"))
        self.screen.blit(pf, (100, 100))
        text = font.render(self.height_text, True, (255, 255, 255))
        self.screen.blit(text, [120, 110])
except:
    pf = pygame.Surface((80, 50))
    pf.fill(pygame.Color("#ff0000"))
    self.screen.blit(pf, (100, 100))
    text = font.render(self.height_text, True, (255, 255, 255))
    self.screen.blit(text, [120, 110])
```

```
            pygame.display.update()

import pygame
import sys
import platform as pl
import ball as b
import map as m
import statistic
import block as bl
import math
import random
import menu
import datetime
import bonus


class Game:
    eps = 1.0
    music_files = {1: "Music/megalovania.mp3",
                   2: "Music/limelight.mp3",
                   3: "Music/tripel.mp3",
                   4: "Music/voiceless.mp3",
                   5: "Music/highscore.mp3",
                   6: "Music/monster.mp3",
                   7: "Music/anthem.mp3",
                   8: "Music/ethereal.mp3",
                   9: "Music/mayday.mp3",
                   10: "Music/medal.mp3"}

    def __init__(self, id=1, score=0, life=3, f=None, map=None):
        self.custom = False
        if map is not None:
            self.current_level_index = "Custom.{0}".format(map)
            self.custom = True
            self.life = life
            self.score = score
            self.multiplier = 1.0
            self.map = m.Map(map)
            self.current_level = self.map.map
            self.field_width = len(self.current_level[0]) * 20 - 20
            self.win_width = self.field_width + 150
```

```
        self.win_height = len(self.current_level) * 20
        self.blocks = self.map.blocks
        self.platform = pl.Platform(self.field_width)
        self.ball = b.Ball(self.field_width, self.win_height)
    elif f is not None:
        args = f.split(";")
        if args[0][:6] == "Custom":
            self.custom = True
        if not self.custom:
            self.current_level_index = int(args[0])
            try:
                self.map = m.Map("Levels/level" +
                                 str(self.current_level_index) + ".txt")
            except:
                stat = statistic.Statistic("{0}"
                                           .format
                                           (self.current_level_index),
                                           self.score)
                stat.draw_stats()
        else:
            self.map = m.Map(args[0][7:])
        self.score = int(args[1])
        self.life = int(args[2])
        self.multiplier = float(args[3])

        self.current_level = self.map.map
        self.field_width = len(self.current_level[0]) * 20 - 20
        self.win_width = self.field_width + 150
        self.win_height = len(self.current_level) * 20
        self.platform = pl.Platform(self.win_width,
                                    float(args[4]), float(args[5]))
        b_args = args[6].split(',')
        self.ball = b.Ball(self.field_width, self.win_height,
                           float(b_args[0]), float(b_args[1]),
                           float(b_args[2]),
                           float(b_args[3]),
                           float(self.win_height - 50),
                           float(b_args[4]))
        self.blocks = []
        for i in range(7, len(args)):
            if args[i] == '':
```

```
                break
            block_args = args[i].split(',')
            self.blocks.append(bl.Block(int(block_args[0]),
                                        int(block_args[1]),
                                        int(block_args[2])))
        else:
            self.current_level_index = id
            self.life = life
            self.score = score
            self.multiplier = 1.0
            try:
                self.map = m.Map("Levels/level" +
                                 str(self.current_level_index) + ".txt")
            except:
                stat = statistic.Statistic(
                    "{0}".format(self.current_level_index - 1),
                    self.score)
                stat.draw_stats()
            self.current_level = self.map.map
            self.field_width = len(self.current_level[0]) * 20 - 20
            self.win_width = self.field_width + 150
            self.win_height = len(self.current_level) * 20
            self.blocks = self.map.blocks
            self.platform = pl.Platform(self.field_width)
            self.ball = b.Ball(self.field_width, self.win_height)
        self.active_bonuses = []
        self.display = (self.win_width, self.win_height)
        self.background_color = "#001a82"
        self.border_color = "#000e47"
        self.on_pause = False
        self.lose = False
        self.screen = pygame.display.set_mode(self.display)
        self.bg = pygame.Surface(self.display)
        self.timer = pygame.time.Clock()
        self.ctrl_pressed = False
        self.ball_cant_drop = False

    def start(self):
        pygame.init()
        pygame.mouse.set_visible(False)
        pygame.display.set_caption("Arkanoid")
```

```
    if not self.custom:
        pygame.mixer.music.load(self.music_files[self.current_level_index])
        pygame.mixer.music.set_volume(0.00)
        pygame.mixer.music.play(25)
    self.bg.fill(pygame.Color(self.background_color))
    while True:
        self.timer.tick(200)
        for e in pygame.event.get():
            if e.type == pygame.QUIT:
                sys.exit()
            self.handle_pressed_keys(e)
        if not self.on_pause:
            self.move_platform()
            self.ball.move()
            self.reflect_ball_by_wall()
            self.reflect_ball_by_block()
            self.move_bonuses()
            self.draw_elements()
            pygame.display.update()
        else:
            self.draw_pause()
            pygame.display.update()

def handle_pressed_keys(self, e):
    if e.type == pygame.KEYDOWN and e.key == pygame.K_ESCAPE:
        pygame.mixer.music.stop()
        menu.Menu()
    if e.type == pygame.KEYDOWN and e.key == pygame.K_a:
        self.platform.MOVING_LEFT = True
    if e.type == pygame.KEYDOWN and e.key == pygame.K_d:
        self.platform.MOVING_RIGHT = True
    if e.type == pygame.KEYUP and e.key == pygame.K_a:
        self.platform.MOVING_LEFT = False
    if e.type == pygame.KEYUP and e.key == pygame.K_d:
        self.platform.MOVING_RIGHT = False
    if e.type == pygame.KEYDOWN and e.key == pygame.K_b:
        self.execute_cheat("destroy block")
    if e.type == pygame.KEYDOWN and e.key == pygame.K_n:
        self.execute_cheat("no lose")
    if e.type == pygame.KEYDOWN and e.key == pygame.K_i:
        self.execute_cheat("decrease speed")
```

```
    if e.type == pygame.KEYDOWN and e.key == pygame.K_o:
        self.execute_cheat("increase speed")
    if e.type == pygame.KEYDOWN and e.key == pygame.K_r:
        self.ball.reincarnate()
        self.eps = 1.0
    if e.type == pygame.KEYDOWN and e.key == pygame.K_q:
        if self.on_pause:
            self.on_pause = False
        else:
            self.on_pause = True
    if e.type == pygame.KEYDOWN and e.key == pygame.K_LCTRL:
        self.ctrl_pressed = True
    if e.type == pygame.KEYUP and e.key == pygame.K_LCTRL:
        self.ctrl_pressed = False
    if e.type == pygame.KEYDOWN and e.key == pygame.K_s:
        if self.ctrl_pressed:
            self.save_game()
    if e.type == pygame.KEYDOWN and e.key == pygame.K_KP_MINUS:
        volume = pygame.mixer.music.get_volume()
        volume -= 0.01
        pygame.mixer.music.set_volume(volume)
    if e.type == pygame.KEYDOWN and e.key == pygame.K_KP_PLUS:
        volume = pygame.mixer.music.get_volume()
        volume += 0.01
        pygame.mixer.music.set_volume(volume)

def save_game(self):
    d = datetime.datetime.now()
    filename = "save-{0}-{1}-{2}-{3}-{4}-{5}" \
        .format(d.year, d.month, d.day, d.hour,
                d.minute, d.second)
    game = str(self.current_level_index) + ";"
    game += str(self.score) + ";"
    game += str(self.life) + ";"
    game += str(self.multiplier) + ";"
    game += str(self.platform.LEFT_COORD) + ";"
    game += str(self.platform.WIDTH) + ";"
    game += str(self.ball.x) + ',' + str(self.ball.y) + ',' + \
        str(self.ball.speed[0]) + ',' + \
        str(self.ball.speed[1]) + ',' + \
        str(self.ball.power) + ";"
```

```
    for block in self.blocks:
        game += str(block.x) + ',' + str(block.y) + ',' + \
                str(block.strength) + ";"
    file = open("Saves/{0}.txt".format(filename), 'w')
    file.write(game)
    file.close()

def execute_cheat(self, cheat):
    # if self.life == 1:
    #     return
    # self.life -= 1
    if cheat == "destroy block":
        index = random.randint(0, len(self.blocks) - 1)
        self.blocks.remove(self.blocks[index])
        self.check_win()
    if cheat == "no lose":
        self.ball_cant_drop = not self.ball_cant_drop
    if cheat == "decrease speed":
        self.ball.speed[0] /= 2
        self.ball.speed[1] /= 2
        self.eps /= 2
    if cheat == "increase speed":
        self.ball.speed[0] *= 2
        self.ball.speed[1] *= 2
        self.eps *= 2

def move_platform(self):
    if self.platform.LEFT_COORD >= 20 and self.platform.MOVING_LEFT:
        self.platform.move(-1)
    if self.platform.RIGHT_COORD <= self.field_width - 20 \
            and self.platform.MOVING_RIGHT:
        self.platform.move(1)

def reflect_ball_by_wall(self):
    if math.fabs(self.ball.left - 20) < self.eps or \
            math.fabs(self.ball.right -
                      (self.field_width - 20)) < self.eps:
        self.ball.speed[0] = -self.ball.speed[0]
        return
    if math.fabs(self.ball.top - 20) < self.eps:
        self.ball.speed[1] = -self.ball.speed[1]
```

```
            return
        if math.fabs(self.ball.bottom - (self.win_height - 40)) < self.eps:
            self.reflect_ball_by_platform()

    def reflect_ball_by_platform(self):
        if self.ball.right < self.platform.LEFT_COORD or \
                self.ball.left > self.platform.RIGHT_COORD:
            self.multiplier = 1.0
            if not self.ball_cant_drop:
                self.eps = 1.0
                self.score -= int(self.score // 5)
                self.life -= 1
                if self.life == 0:
                    pygame.mixer.music.stop()
                    if not self.custom:
                        stats = statistic.Statistic(
                            str(self.current_level_index - 1), self.score)
                        stats.draw_stats()
                    menu.Menu()
                else:
                    self.ball.reincarnate()
            else:
                self.ball.speed[1] = -self.ball.speed[1]
            return
        self.score += int(10 * self.multiplier)
        self.multiplier = 1.0
        if self.ball.x < self.platform.LEFT_COORD:
            self.ball.speed[0] = -self.ball.basic_speed
            self.ball.speed[1] = -self.ball.basic_speed
        elif self.ball.x > self.platform.RIGHT_COORD:
            self.ball.speed[0] = self.ball.basic_speed
            self.ball.speed[1] = -self.ball.basic_speed
        else:
            middle = self.platform.WIDTH // 2
            pos = self.ball.x - self.platform.LEFT_COORD
            if pos < middle:
                angle = -1 + (pos / middle)
                self.ball.speed[0] = angle
            else:
                angle = (pos / middle) - 1
                self.ball.speed[0] = angle
```

```
            self.ball.speed[1] = \
                -math.sqrt(2 - math.pow(self.ball.speed[0], 2))
        print(self.ball.speed)

    def reflect_ball_by_block(self):
        for block in self.blocks:
            if math.fabs(self.ball.top - block.bottom) < self.eps or \
                    math.fabs(self.ball.bottom - block.top) < self.eps:
                if block.left <= self.ball.left <= block.right or \
                        block.left <= self.ball.right <= block.right:
                    self.ball.speed[1] = -self.ball.speed[1]
                    self.score += int(20 * self.multiplier)
                    if block.decrease_and_check_destroying(self.ball.power):
                        self.score += int(100 * self.multiplier)
                        if block.bonus is not None:
                            if block.bonus == "destroy_line":
                                destr = []
                                for b in self.blocks:
                                    if b.y == block.y and b != block:
                                        destr.append(b)
                                self.score += len(destr) * 150
                                for d in destr:
                                    self.blocks.remove(d)
                                destr.clear()
                            else:
                                bon = bonus.Bonus(
                                    block.bonus, block.left, block.top)
                                self.active_bonuses.append(bon)
                        self.blocks.remove(block)
                    self.multiplier += 0.1
                    self.check_win()
                    return
            elif math.fabs(self.ball.left - block.right) < self.eps or \
                    math.fabs(self.ball.right - block.left) < self.eps:
                if block.top <= self.ball.top <= block.bottom or \
                        block.top <= self.ball.bottom <= block.bottom:
                    self.ball.speed[0] = -self.ball.speed[0]
                    self.score += int(20 * self.multiplier)
                    if block.decrease_and_check_destroying(self.ball.power):
                        self.score += int(100 * self.multiplier)
                        if block.bonus is not None:
```

```
                    if block.bonus == "destroy_line":
                        destr = []
                        for b in self.blocks:
                            if b.y == block.y and b != block:
                                destr.append(b)
                        self.score += len(destr) * 150
                        for d in destr:
                            self.blocks.remove(d)
                        destr.clear()
                    else:
                        bon = bonus.Bonus(
                            block.bonus, block.left, block.top)
                        self.active_bonuses.append(bon)
                self.blocks.remove(block)
            self.multiplier += 0.1
            self.check_win()
            return

    def move_bonuses(self):
        if len(self.active_bonuses) > 0:
            for bon in self.active_bonuses:
                bon.y += bon.speed[1]
                if bon.y == self.win_height - 40:
                    if self.platform.LEFT_COORD <= bon.x <= \
                            self.platform.RIGHT_COORD:
                        if bon.name == "powerup":
                            self.ball.power *= 2
                        if bon.name == "platform_more":
                            self.platform.WIDTH = self.platform.WIDTH // 2 * 3
                            self.platform.RIGHT_COORD = \
                                self.platform.LEFT_COORD + self.platform.WIDTH
                        if bon.name == "platform_less":
                            self.platform.WIDTH //= 2
                            self.platform.RIGHT_COORD = \
                                self.platform.LEFT_COORD + self.platform.WIDTH
                    else:
                        self.active_bonuses.remove(bon)

    def check_win(self):
        if len(self.blocks) == 0:
            pygame.mixer.music.stop()
```

```
            if not self.custom:
                g = Game(self.current_level_index + 1,
                         self.score, self.life + 1)
                g.start()
                self.timer = None
            menu.Menu()

    def draw_elements(self):
        self.screen.blit(self.bg, (0, 0))
        self.platform.draw(self.screen, self.win_height)
        x = y = 0
        for row in self.current_level:
            for col in row:
                if col == "B":
                    pf = pygame.Surface((20, 20))
                    pf.fill(pygame.Color(self.border_color))
                    self.screen.blit(pf, (x, y))
                x += 20
            y += 20
            x = 0
        for block in self.blocks:
            block.draw(self.screen)
        pf = pygame.Surface((20, 20))
        pf.fill(pygame.Color(self.ball.color))
        self.screen.blit(pf, (self.ball.x - 10, self.ball.y - 10))
        if len(self.active_bonuses) > 0:
            for bon in self.active_bonuses:
                self.screen.blit(bon.image, (bon.x, bon.y))
        if self.ball_cant_drop:
            pf = pygame.Surface((self.field_width - 40, 2))
            pf.fill(pygame.Color("#ffff00"))
            self.screen.blit(pf, (20, self.win_height - 40))

        font = pygame.font.Font(None, 25)
        lvl = ""
        if self.custom:
            lvl = "Level: Custom"
        else:
            lvl = "Level: {0}".format(self.current_level_index)

        text = font.render(lvl, True, (255, 255, 255))
```

```
        self.screen.blit(text, [self.win_width - 140, 15])
        text = font.render("Score: {0}"
                           .format(self.score), True, (255, 255, 255))
        self.screen.blit(text, [self.win_width - 140, 35])
        text = font.render("Multiplier: {0}"
                           .format(self.multiplier), True, (255, 255, 255))
        self.screen.blit(text, [self.win_width - 140, 55])
        text = font.render("Balls: {0}"
                           .format(self.life), True, (255, 255, 255))
        self.screen.blit(text, [self.win_width - 140, 75])

    def draw_pause(self):
        if self.lose:
            t = "Final score: {0}".format(self.score)
        else:
            t = "Game paused"
        font = pygame.font.Font(None, 45)
        text = font.render(t, True, (255, 0, 0))
        self.screen.blit(text, [self.field_width // 2 - 60,
                                self.win_height // 2])

import block


class Map:

    def __init__(self, filename):
        self.map_in_text = open(filename)
        self.map = []
        self.blocks = []
        self.create_map_from_file()
        self.map_in_text.close()

    def create_map_from_file(self):
        self.map = []
        for line in self.map_in_text:
            self.map.append(line)
        self.blocks = []
        i = j = 0
        for row in self.map:
            for col in row:
```

```
                if col.isdigit():
                    b = block.Block(i * 20 + 10, j * 20 + 10, int(col))
                    self.blocks.append(b)
                i += 1
            i = 0
            j += 1

import pygame
import game
import sys
import editor_info
import select_custom_level
import select_save


class Menu:

    def __init__(self):
        self.display = (220, 120)
        self.screen = pygame.display.set_mode(self.display)
        self.bg = pygame.Surface(self.display)
        self.background_color = "#14005e"
        self.item1 = "1: Start new game"
        self.item2 = "2: Load game"
        self.item3 = "3: Create new level"
        self.item4 = "4: Browse created levels"
        self.timer = pygame.time.Clock()
        self.draw_main_menu()

    def draw_main_menu(self):
        pygame.init()
        pygame.display.set_caption("Main menu")
        self.bg.fill(pygame.Color(self.background_color))
        while True:
            self.timer.tick(200)
            self.screen.blit(self.bg, (0, 0))
            font = pygame.font.Font(None, 25)
            text = font.render(self.item1, True, (255, 255, 255))
            self.screen.blit(text, [10, 15])
            text = font.render(self.item2, True, (255, 255, 255))
            self.screen.blit(text, [10, 40])
```

```
            text = font.render(self.item3, True, (255, 255, 255))
            self.screen.blit(text, [10, 65])
            text = font.render(self.item4, True, (255, 255, 255))
            self.screen.blit(text, [10, 90])
            for e in pygame.event.get():
                if e.type == pygame.QUIT:
                    sys.exit()
                self.handle_keys(e)
            pygame.display.update()

    def handle_keys(self, e):
        if e.type == pygame.KEYDOWN and e.key == pygame.K_ESCAPE:
            sys.exit(0)
        elif e.type == pygame.KEYDOWN and e.key == pygame.K_1:
            g = game.Game(id=1, score=0, life=3)
            g.start()
        elif e.type == pygame.KEYDOWN and e.key == pygame.K_2:
            select_save.SaveSelector()
        elif e.type == pygame.KEYDOWN and e.key == pygame.K_3:
            editor_info.EditorMapInfo()
        elif e.type == pygame.KEYDOWN and e.key == pygame.K_4:
            select_custom_level.CustomLevelSelector()
        if e.type == pygame.KEYDOWN and e.key == pygame.K_ESCAPE:
            sys.exit(0)

import pygame as pg


class Platform:

    def __init__(self, screen_width, left=-1, width=-1):
        self.MOVING_LEFT = False
        self.MOVING_RIGHT = False
        self.COLOR = "#789ABC"
        self.HEIGHT = 20
        if width == -1:
            self.WIDTH = screen_width / 4
        else:
            self.WIDTH = width
        if left == -1:
            self.LEFT_COORD = screen_width / 2 - self.WIDTH / 2
```

```
        else:
            self.LEFT_COORD = left
        self.RIGHT_COORD = self.LEFT_COORD + self.WIDTH
        self.MOVE_SPEED = screen_width / 250

    def move(self, rotation):
        self.LEFT_COORD += rotation * self.MOVE_SPEED
        self.RIGHT_COORD += rotation * self.MOVE_SPEED

    def draw(self, screen, height):
        pf = pg.Surface((self.WIDTH, 20))
        pf.fill(pg.Color(self.COLOR))
        screen.blit(pf, (self.LEFT_COORD, height - 40))

class Player:

    def __init__(self, name, levels, score):
        self.name = name
        self.levels = levels
        self.score = score

    def __str__(self):
        return "{0}/{1}/{2}".format(self.name, self.levels, self.score)

import pygame
import player
import sys
import menu


class RecordsScreen:

    def __init__(self, index):
        self.player_index = index
        f = open("records.txt", 'r')
        self.players = []
        for line in f:
            args = line.split('/')
            self.players.append(player.Player(args[0], args[1], int(args[2])))
        f.close()
        self.display = (400, 400)
```

```
        self.timer = pygame.time.Clock()
        self.screen = pygame.display.set_mode(self.display)
        self.bg = pygame.Surface(self.display)
        self.draw()

    def draw(self):
        pygame.init()
        pygame.display.set_caption("Records")
        while True:
            self.timer.tick(200)
            self.screen.blit(self.bg, (0, 0))
            font = pygame.font.Font(None, 25)
            text = font.render("Name", True, (255, 255, 255))
            self.screen.blit(text, [10, 15])
            text = font.render("Levels", True, (255, 255, 255))
            self.screen.blit(text, [150, 15])
            text = font.render("Score", True, (255, 255, 255))
            self.screen.blit(text, [250, 15])
            x = 50
            for i in range(0, min(10, len(self.players))):
                color = (255, 255, 255)
                if i == self.player_index:
                    color = (0, 255, 0)
                text = font.render(self.players[i].name, True, color)
                self.screen.blit(text, [10, x])
                text = font.render(self.players[i].levels, True, color)
                self.screen.blit(text, [150, x])
                text = font.render(str(self.players[i].score), True, color)
                self.screen.blit(text, [250, x])
                x += 25
            for e in pygame.event.get():
                if e.type == pygame.QUIT:
                    sys.exit()
                if e.type == pygame.KEYDOWN and e.key == pygame.K_ESCAPE:
                    menu.Menu()
            pygame.display.update()

import pygame
import os
import sys
import game
```

```
import menu


class CustomLevelSelector:
    def __init__(self):
        self.levels = os.listdir("./CreatedLevels")
        self.cursor = 0
        height = 30
        if len(self.levels) > 0:
            height = len(self.levels) * 20
        self.display = (320, height)
        self.screen = pygame.display.set_mode(self.display)
        self.bg = pygame.Surface(self.display)
        self.timer = pygame.time.Clock()
        self.draw()

    def draw(self):
        pygame.init()
        pygame.display.set_caption("Created levels")
        self.bg.fill(pygame.Color("#14005e"))
        while True:
            self.timer.tick(200)
            self.screen.blit(self.bg, (0, 0))
            for e in pygame.event.get():
                if e.type == pygame.QUIT:
                    sys.exit()

                if e.type == pygame.KEYDOWN and e.key == pygame.K_ESCAPE:
                    menu.Menu()

                if e.type == pygame.KEYDOWN and e.key == pygame.K_UP:
                    if 0 < self.cursor:
                        self.cursor -= 1

                if e.type == pygame.KEYDOWN and e.key == pygame.K_DOWN:
                    if self.cursor < len(self.levels) - 1:
                        self.cursor += 1

                if e.type == pygame.KEYDOWN and e.key == pygame.K_RETURN:
                    g = game.Game(map="CreatedLevels/" +
                                  self.levels[self.cursor])
```

```
                g.start()

            if e.type == pygame.KEYDOWN and e.key == pygame.K_DELETE:
                delete = self.levels[self.cursor]
                self.levels.remove(delete)
                os.remove("./CreatedLevels/" + delete)
        font = pygame.font.Font(None, 20)
        if len(self.levels) == 0:
            text = font.render("No levels", True, (255, 0, 0))
            self.screen.blit(text, [10, 10])
        else:
            y = 5
            for i in range(0, len(self.levels)):
                color = (255, 255, 255)
                if i == self.cursor:
                    color = (0, 255, 0)
                text = font.render(self.levels[i], True, color)
                self.screen.blit(text, [10, y])
                y += 20

        pygame.display.update()
import pygame
import os
import sys
import game
import menu


class SaveSelector:
    def __init__(self):
        self.saves = os.listdir("./Saves")
        self.cursor = 0
        height = 30
        if len(self.saves) > 0:
            height = len(self.saves) * 20
        self.display = (320, height)
        self.screen = pygame.display.set_mode(self.display)
        self.bg = pygame.Surface(self.display)
        self.timer = pygame.time.Clock()
        self.draw()
```

```
def draw(self):
    pygame.init()
    pygame.display.set_caption("Saves")
    self.bg.fill(pygame.Color("#14005e"))
    while True:
        self.timer.tick(200)
        self.screen.blit(self.bg, (0, 0))
        for e in pygame.event.get():
            if e.type == pygame.QUIT:
                sys.exit()

            if e.type == pygame.KEYDOWN and e.key == pygame.K_ESCAPE:
                menu.Menu()

            if e.type == pygame.KEYDOWN and e.key == pygame.K_UP:
                if 0 < self.cursor:
                    self.cursor -= 1

            if e.type == pygame.KEYDOWN and e.key == pygame.K_DOWN:
                if self.cursor < len(self.saves) - 1:
                    self.cursor += 1

            if e.type == pygame.KEYDOWN and e.key == pygame.K_RETURN:
                saved = open("Saves/" + self.saves[self.cursor], 'r')
                g = game.Game(f=saved.read())
                saved.close()
                g.start()

            if e.type == pygame.KEYDOWN and e.key == pygame.K_DELETE:
                delete = self.saves[self.cursor]
                self.saves.remove(delete)
                os.remove("./Saves/" + delete)
        font = pygame.font.Font(None, 20)
        if len(self.saves) == 0:
            text = font.render("No saves", True, (255, 0, 0))
            self.screen.blit(text, [10, 10])
        else:
            y = 5
            for i in range(0, len(self.saves)):
                color = (255, 255, 255)
                if i == self.cursor:
```

```
                color = (0, 255, 0)
            text = font.render(self.saves[i], True, color)
            self.screen.blit(text, [10, y])
            y += 20

        pygame.display.update()

import pygame
import sys
import player
import records_screen


class Statistic:

    def __init__(self, passed, score):
        self.passed = passed
        self.score = score
        self.display = (320, 240)
        self.timer = pygame.time.Clock()
        self.screen = pygame.display.set_mode(self.display)
        self.bg = pygame.Surface(self.display)
        f = open("records.txt", 'r')
        self.records = []
        for line in f:
            args = line.split('/')
            self.records.append(player.Player(args[0], args[1], int(args[2])))
        f.close()

    def draw_stats(self):
        pygame.init()
        pygame.display.set_caption("Statistic")
        name = ""
        while True:
            self.timer.tick(200)
            self.screen.blit(self.bg, (0, 0))
            font = pygame.font.Font(None, 25)
            text = font.render("You have passed {0} levels"
                               .format(self.passed), True, (255, 255, 255))
            self.screen.blit(text, [10, 15])
            text = font.render("Your final score: {0}"
```

```
                .format(self.score), True, (255, 255, 255))
        self.screen.blit(text, [10, 40])
        text = font.render("Enter your name", True, (255, 255, 255))
        self.screen.blit(text, [10, 65])
        text = font.render(name, True, (0, 0, 255))
        self.screen.blit(text, [10, 90])
        for e in pygame.event.get():
            if e.type == pygame.QUIT:
                sys.exit()
            if e.type == pygame.KEYDOWN and e.key == pygame.K_RETURN:
                pl = player.Player(name, self.passed, self.score)
                self.records.append(pl)
                self.records.sort(key=lambda x: x.score, reverse=True)
                f = open("records.txt", 'w')
                f.truncate()
                for p in self.records:
                    f.write(str(p) + "\n")
                index = self.records.index(pl)
                f.close()
                r = records_screen.RecordsScreen(index)
            if e.type == pygame.KEYDOWN:
                if e.key == pygame.K_BACKSPACE:
                    name = name[:-1]
                else:
                    name += e.unicode
                text = font.render(name, True, (0, 0, 255))
                self.screen.blit(text, [10, 90])
        pygame.display.update()
```

Заключение

В данной курсовой работе была разработана игра “Arkanoid”. Для реализации данного проекта был использован язык программирования Python, так как использование его преимуществ (инкапсуляция, полиморфизм, наследование) позволяет сделать код понятнее и не перегружать его лишними элементами. Пользовательский интерфейс был спроектирован при помощи библиотеки PyGame. В процессе разработки были изучены особенности и основные команды, используемые в библиотеке. Также были решены все поставленные в начале разработки задачи, а именно: 1 Изучение принципов работы с библиотекой PyGame. 2 Проектирование графического пользовательского интерфейса. 3 Закрепление и углубление знаний и навыков, полученных при изучении дисциплины “Объектно-ориентированного программирование”. В качестве перспектив развития проекта можно указать добавление более

высокое графическое оформление, настройка уровня сложности, добавление врагов. Также необходимо отметить, что проект был занесён на онлайн-репозиторий GitHub. Исходя из всего вышесказанного, можно сделать вывод, что тема данной курсовой работы является достаточно актуальной в связи с быстрым развитием технологий и их возрастающей роли в нашей жизни. Вскоре не останется ни одной сферы, в которой бы они не были бы задействованы

Список использованных источников

1. [url] **Основы Git** https://git-scm.com/
2. [url] **Введение в MVC и HMVC** https://ruseller.com/
3. [url] **Паттерны проектирования** https://habr.com/ru/company/otus/blog/451516/.
4. [url] **PyGame — шпаргалка для использования** https://waksoft.susu.ru/2019/04/24/pygame-shpargalka-dlja-ispolzovanija/
5. [url] **PyGame** https://www.pygame.org/news
6. [url] **Объектно-ориентированное Программирование в Python** https://python-scripts.com/object-oriented-programming-in-python

Приложения

1. [электронный документ] 5ebd10a848e4d_Записка.docx