**Паспорт проекта**

**Название:** “Арканоид”

**Авторы проекта:** Степан Фёдоров, Леонид Кузякин

**Описание идеи.**

Классическая игра арканоид.

**Описание реализации:**

**Класс Board** - основной класс, который отвечает за игровой процесс.

Для реализации класса использовались многие возможности pygame'а, а именно: звуковые эффекты, спрайты и другое.

Кроме того был задействован модуль random для случайного создания шарика, framework для работы с ивентами и передачей данных между сценами.

И load\_image для загрузки png картинок.

Как уже было упомянуто, данный класс отвечает за сам игровой процесс.

То есть за движение шарика по игровому полю, движение кирпичей и игровой доски, рендер всего этого и многое другое.

Управление игровой доской происходит за счёт движения мыши по окну игры.

Напомню, что у нас реализован классический Арканоид, то есть шарик скачет по игровому полю, уничтожая собой кирпичи.

Есть 3 жизни, 3 уровня сложности, подсчёт очков и многое другое.

**Классы сцен Designer, Customization, Difficulty** - создание уровней, выбор спрайтов на игровом поле и выбор уровня сложности.

Для реализации всех перечисленных классов использовались возможности pygame'а, framework'а и load\_image'а.

Designer позволяет создать уровень, в который можно будет сразу же сыграть.

Если же уровень не был создан игроком, то он будет играть в стандартный, заранее созданный.

Customization даёт возможность изменить цвет кирпичей и доски на игровом поле.

Difficulty - элементарный выбор уровня сложности, от которого будет зависеть с какой скоростью будет лететь шарик.

В процессе разработки было обнаружено много повторяющегося кода.

Для облегчения читаемости, уменьшения количества багов и увеличения скорости разработки был создан модуль “**framework**” предоставляющий классы и функции для удобного и качественногосоздания игры.

Для удобной обработки событий был создан декоратор @**event\_handler**.

Благодаря ему не требуется раздувать основной цикл приложения и создавать сложную логику.

Для удобства разделения приложения на отдельные файлы и классы было решено создавать свой класс для каждой “сцены” в приложении.

В нашем случае сцена - элементы на экране и логика и работа. К примеру меню - это одна сцена, а форма регистрации - другая.

Для оркестрации сцен и переключения между ними используется класс **SceneManager**.

В приложении неоднократно приходилось использовать кнопки и поля для ввода текста.

Для упрощения их создания вся логика работы с ними была вынесена в отдельные компоненты - **Button и TextEdit** соответственно.

Для работы с базой данных используются **Storage** классы - **UserStorage и LeaderboardStorage**.

Они скрывают в себе низкоуровневую логику работы с базой данной, такую как, например, SQL запросы, и предоставляют простой и понятный интерфейс.