**Пояснительная записка**

**Название проекта**: «Самое ценное сокровище»

**Исполнитель:** Иванова Юлия Александровна, Русанов Андрей Сергеевич

**Цель проекта:** Создать игру-платформер на 2 персонажей.

**Используемые библиотеки:**

* + **pygame**
  + **sqlite3 3.3.3**
  + **pip 24.3.1**
  + **os**
  + **sys**

**План проекта:**

1. ***Создание приветственного экрана***
   1. Создание экрана с уровнями
2. ***Создание уровней для игры***
   1. Создание сюжета и концепции игры
   2. Создание Персонажей
   3. Создание уровней
   4. Спрайты
3. ***Украшения игры***
4. ***Скриншоты приложения***
5. ***База данных***
6. ***Используемые функции и классы***

**Необходимые таблицы в базе данных:**

1. users (Поля: levelOne\_complite, levelTwo\_complite, levelThree\_complite, levelFour\_complite, total\_complite)

Хранение количества пройденных уровней пользователем.

1. levelOne (Поля: time, coins)

Хранение статистики о прохождении уровня 1

1. levelTwo (Поля: time, coins)

Хранение статистики о прохождении уровня 2

1. levelThree (Поля: time, coins)

Хранение статистики о прохождении уровня 3

1. levelFour (Поля: time, coins)

Хранение статистики о прохождении уровня 4

**1.Создание приветственного экрана**

Приветственный экран включает в себя: кнопку для запуска и для просмотра правил игры. (рисунок 1)

**1.1 Создание экрана с уровнями**

Экран с уровнями включает в себя 5 кнопок. Четыре кнопки для входа на уровни (пока не пройден предыдущий уровень, следующий не открывается) и кнопка exit для выхода в главное меню. (2 рисунок)

**2.Создание уровней для игры**

**2.1 Создание сюжета и концепции игры**

**Описание:**

Маг и разбойник находят карту сокровищ и отправляются на их поиски. Игра выполнена в формате платформера и рассчитана на двух игроков. У каждого из двух персонажей есть свои особенности:

*Маг (рисунок 3):*

* Способен открывать магические двери
* Низкого роста
* Высоко прыгает
* Не может убивать монстров

*Разбойник (рисунок 4):*

* Высокого роста
* Низко прыгает
* Может убивать монстров
* Может двигать коробки

Для того чтобы пройти игру, игрокам необходимо работать вместе.

**2.2 Создание Персонажей**

Каждый спрайт в игре является наследником от класса GameSprite. Полями этого класса являются: положение спрайта по x и y, группа спрайтов к которой спрайт принадлежит, картинка, ширина и высота спрайта. Все спрайты, которые могут двигаться представлены классом MovableGameSprite, который в свою очередь является наследником класса AnimatedSprite. Этот класс отвечает за анимацию у спрайтов. Является наследником класса GameSprite.

Персонаж маг представлен классом Mag. Этот класс является наследником класса MovableGameSprite. У класса мага есть функции: can\_move (Проверяет может ли персонаж пройти дальше), сan\_stay (проверяет может ли персонаж стоять на данном блоке), do\_update (отвечает за обработку всех событий, связанных с персонажем), а также несколько функций, отвечающих за анимацию. Класс разбойника создан аналогично.

**2.3 Создание Уровней**

Каждый уровень представлен в виде текстового файла. При запуске программа читает из него расположение каждого спрайта и добавляет их на экран. На уровне есть монетки, их необходимо собирать. Для того чтобы уровень засчитался пройденным необходимо обоим персонажам достичь дверей, расположенных на уровне. В базу данных будет заноситься информация о лучшем прохождении (сколько времени затрачено, сколько монет собрано).

В игре 4 уровня (каждый из них представлен в приложении). Уровень представлен одним из классов LevelOne (рисунок 5), LevelTwo (рисунок 6), LevelThree (рисунок 7), LevelFour (рисунок 8) соответственно. Все являются наследником от класса Level. Они отвечают за отрисовку уровня, а также за обработку его событий. После проигрыша или победы запускается экран сообщающий это. Так же после победы подсчитываются результаты. Лучшие заносятся в базу данных.

**2.4 Спрайты**

Помимо двух основных в игре есть 11 различных спрайтов. Все они представлены классами, которые являются наследниками от GameSprite или MovableGameSprite.

**Class Block** (рисунок 9):

Является наследником от класса GameSprite. Представляет собой обычный недвижимый блок. Персонажи не могут ходить сквозь него. Особых функций нет

**Class Coin** (рисунок 10):

Является наследником от класса GameSprite. Представляет собой монету. Персонажи могут их собирать. Есть функция, которая отвечает за подсчёт количества собранных монет и их удаление.

**Class Door** (рисунок 11):

Является наследником от класса GameSprite. Представляет собой дверь. Её необходимо достичь, чтобы пройти уровень. Есть функция, которая проверяет наличие персонажа рядом.

**Class MagicDoor** (рисунок 12):

Является наследником от класса GameSprite. Представляет собой особое препятствие. Персонажи не могут пройти сквозь него, однако маг может открывать такие двери. Есть функция, которая отвечает за эту механику.

**Class Button** (рисунок 13):

Является наследником от класса GameSprite. Представляет собой кнопку, которая открывает некоторые двери. Чтобы её активировать на неё необходимо поставить коробку или подойти к ней и зажать определённую кнопку. Есть функция, которая проверяет условия для активации.

**Class Lever** (рисунок 14):

Является наследником от класса GameSprite. Представляет собой рычаг, который открывает некоторые двери. Чтобы его активировать нужно подойти к нему и нажать определённую кнопку. Есть функция, которая проверяет условия для активации.

**Class GorizontalDoor и Class VerticalDoor** (рисунок 15 и 16):

Оба класса несут в себе похожие функции, единственное отличие — это их поворот (объекты одного класса горизонтальной ориентации, а другого вертикальной). Являются наследником от класса GameSprite. Представляют собой дверь или люк, который открывается только после взаимодействия с рычагом или кнопкой. Есть две функции: connect и check. Первая отвечает за привязку определенной кнопки или рычага к двери или люку, а вторая за проверку состояния привязанной кнопки или рычага. Если параметр activate у привязанного объекта равен True, то дверь открывается.

**Class Box** (рисунок 17):

Является наследником от класса MovableGameSprite, но также для работы использует класс BoxService (Из-за того, что коробка обладает свойствами движимого и недвижимого объектов, для неё был написан дополнительный класс. Он корректирует дистанцию, которую проходит персонаж до столкновения с коробкой или падения на неё). Представляет собой блок, на котором можно стоять. Так же его может передвигать разбойник. При помощи коробки можно активировать кнопки.

**Class Spike** (рисунок 18):

Является наследником от класса GameSprite. Представляет собой шипы, при столкновении с которыми персонаж умирает. Есть функция, которая отслеживает столкновение игрового спрайта с шипами.

**Class Monsters** (рисунок 19):

Является наследником от класса MovableGameSprite. Представляет собой монстра, который при столкновении с игроком убивает его (Только разбойник способен убивать монстров). Есть функции необходимые для движения, а также для проверки столкновения персонажа и монстра.

**3.Украшение игры**

**Анимации:**

Представлены классом AnimatedSprite. Для того чтобы добавить анимацию используется функция add\_animation. На вход она принимает список с номерами кадров анимации, которые необходимо показывать. Причем если номер кадра будет находится за пределами диапазона, то будут показываться все кадры подряд. Так же в классе MovableGameSprite есть функции, которые возвращают номера анимаций, которые необходимо проиграть в той или иной ситуации.

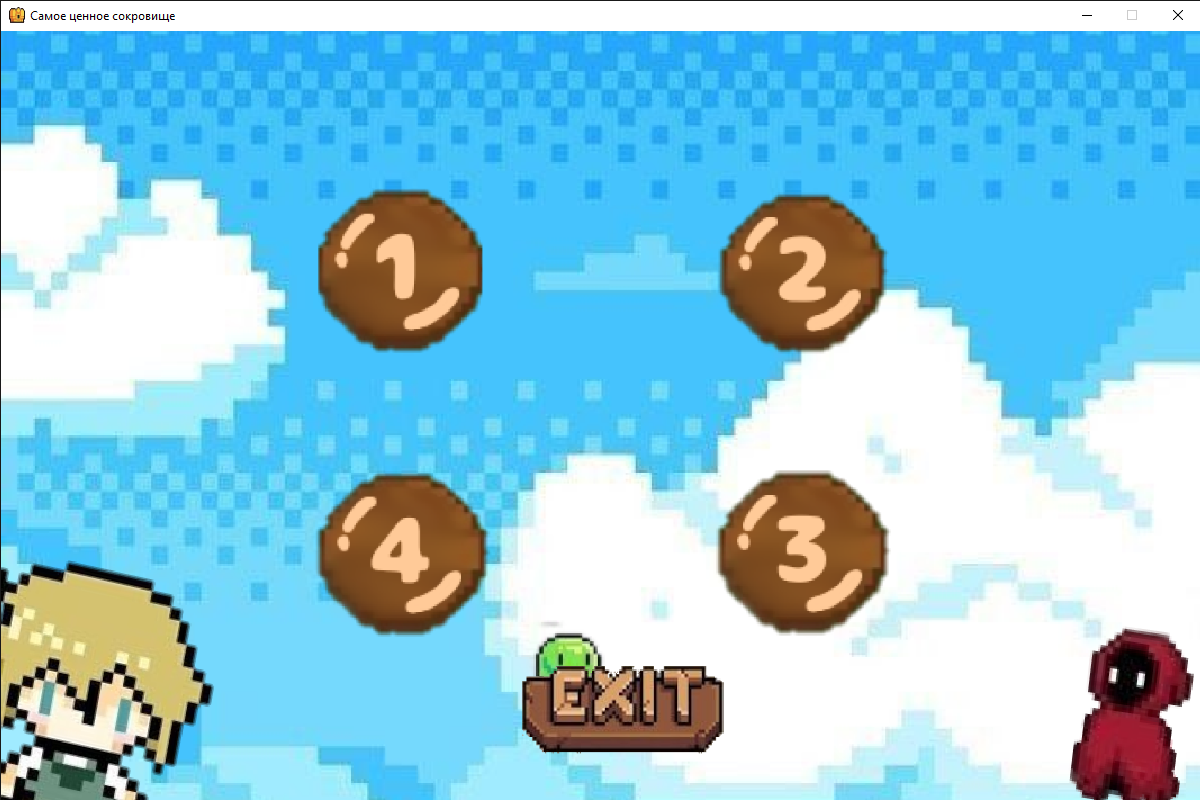
**Звуки:**

Для удобства использования звуки загружаются в отдельном файле. Реализованы они при помощи библиотеки pygame.mixer.

**4.Скриншоты приложения**



Рисунок



Рисунок



Рисунок



Рисунок



Рисунок



Рисунок



Рисунок



Рисунок

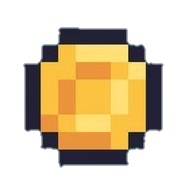
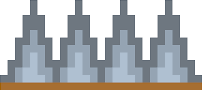
   

Рисунок Рисунок 10 Рисунок 11 Рисунок 12 Рисунок 13 Рисунок 14

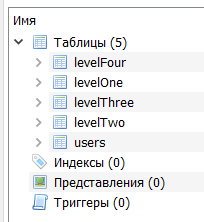
   

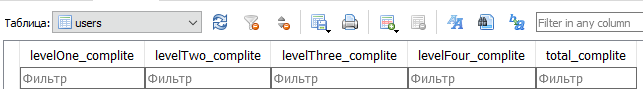
*Рисунок 15 Рисунок 16 Рисунок 17 Рисунок 18*

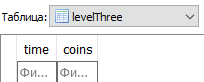
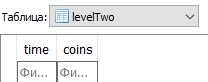
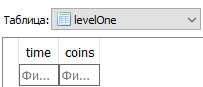


Рисунок 19

**5.База данных**







**6.Используемые функции и классы**

**class Board:**

* set\_view – устанавливает размер клетки и их количество в клетчатом поле.
* render – размечает клетчатое поле.
* Coordinates – возвращает координаты клетки.

**class BegingGame:**

* start – запускает начальную катсцену

**class Levels:**

* start – запускает экран с уровнями, а так же отслеживает нажатия пользователя.

**class Level:**

* init – готовит всё необходимое для запуска игрового цикла.
* Initialization – добавляет все спрайты необходимые для уровня в группы.
* correct\_levelMap – корректирует карту уровня.
* bild – рисует уровень.
* display\_message - отображение сообщения на экране.
* show\_win\_screen - показ окна победы.
* show\_resalt\_screen - показ окна результатов.
* show\_lose\_screen - показ окна поражения.

**class LevelOne:**

* paint – запускает первый уровень.

**class LevelTwo:**

* paint – запускает второй уровень.

**class LevelThree:**

* paint – запускает третий уровень.

**class LevelFour:**

* paint – запускает четвёртый уровень.

**class GameSprite:**

* get\_left\_cell\_x – возвращает координату левого угла спрайта в клетках.
* get\_right\_cell\_x – возвращает координату правого угла спрайта в клетках.
* get\_top\_cell\_y - возвращает координату верхнего угла спрайта в клетках.
* get\_bottom\_cell\_y - возвращает координату нижнего угла спрайта в клетках.
* set\_left\_cell\_x – выравнивает левый край спрайта по переданной клетке
* set\_right\_cell\_x – выравнивает правый край спрайта по переданной клетке
* set\_top\_cell\_y – выравнивает верхний край спрайта по переданной клетке
* set\_bottom\_cell\_y – выравнивает нижний край спрайта по переданной клетке

**class AnimatedSprite:**

* add\_animation – добавляет анимацию.
* play\_animation – проигрывает выбранную анимации.
* \_cut\_sheet – нарезает картинку на кадры
* transform\_frame – при необходимости разворачивает кадр в другую сторону.
* update – проигрывает анимацию.

**class MovableGameSprite:**

* change\_direction – меняет направление спрайта.
* set\_direction – устанавливает направление спрайта.
* transform\_frame – при необходимости разворачивает картинку спрайта в другую сторону.
* get\_stay\_animation – возвращает номер анимации бездействия.
* get\_move\_animation – возвращает номер анимации ходьбы.
* get\_jump\_animation – возвращает номер анимации прыжка.
* get\_fall\_animation – возвращает номер анимации падения.
* can\_move – проверяет может ли персонаж двигаться.
* can\_stay – проверяет может ли спрайт стоять на блоках.
* can\_jump\_through – проверяет может ли спрайт прыгать сквозь блоки.
* fall – отвечает за падение спрайта.
* \_process\_jump – вспомогательная функция для прыжка.
* jump – отвечает за прыжок спрайта.
* move – отвечает за движение спрайта.

**Class Block**

**class Coin:**

* update – отвечает за механику сбора монет.

**class Door:**

* update – проверяет находится ли рядом персонаж.
* check\_sprite\_inside - проверяет, находится ли спрайт большей частью внутри двери.

**class MagicDoor:**

* update – отвечает за проверку условий для открытия двери.

**class Button:**

* update – отвечает за проверку условий для активации.
* check\_sprite\_inside - проверяет, находится ли спрайт большей частью внутри кнопки.

**class Lever:**

* update – отвечает за проверку условий для активации.
* check\_sprite\_inside - проверяет, находится ли спрайт большей частью внутри рычага.

**class GorizontalDoor:**

* connect - отвечает за привязку определенной кнопки или рычага к люку.
* check – отвечает за проверку состояния привязанной кнопки или рычага.

**class VerticalDoor:**

* connect - отвечает за привязку определенной кнопки или рычага к двери.
* check – отвечает за проверку состояния привязанной кнопки или рычага.

**class Box:**

* can\_move – проверяет, может ли спрайт пройти сквозь блоки.
* can\_stay – проверяет, может ли спрайт стоять на блоках.

**class Spike:**

* update – проверяет столкновение спрайта с шипами.

**class Monsters:**

* can\_move – проверяет, может ли спрайт пройти сквозь блоки.
* can\_stay – проверяет, может ли спрайт стоять на блоках.
* do\_update – отвечает за движение спрайта и его механики.

**class Mag:**

* can\_move – проверяет, может ли спрайт пройти сквозь блоки.
* can\_stay – проверяет, может ли спрайт стоять на блоках.
* get\_stay\_animation – возвращает номер анимации бездействия.
* get\_move\_animation – возвращает номер анимации ходьбы.
* get\_jump\_animation – возвращает номер анимации прыжка.
* get\_fall\_animation – возвращает номер анимации падения.
* do\_update – отвечает за движение спрайта и его механики.

**class Robber:**

* can\_move – проверяет, может ли спрайт пройти сквозь блоки.
* can\_stay – проверяет, может ли спрайт стоять на блоках.
* get\_stay\_animation – возвращает номер анимации бездействия.
* get\_move\_animation – возвращает номер анимации ходьбы.
* do\_update – отвечает за движение спрайта и его механики.

**class BoxService:**

* addBox – добовляет коробку в список.
* moveBoxes – отвечает за передвижение коробки
* stay\_on\_box – проверяет, стоит ли персонаж на коробке.
* fallThroughBoxes – корректирует дистанцию персонажа, если тот падает на коробку (это необходимо потому, что коробка является движимым блоком, в следствии чего её невозможно отследить через карту уровня)

**class EndGame:**

* start – запускает конечную катсцену.

**class Ruls:**

* start – запускает правила игры.

**class mutableInt**

**Вспомогательные функции:**

* load\_image – загружает картинку.
* checkList – проверяет, активирована ли хотя бы одна кнопка из списка.
* get\_db\_structure – возвращает структуру базы данных.
* validate\_and\_fix\_data – отвечает за работу с базой данных.
* create\_database – создаёт базу данных.
* reset\_database – удаляет данные из базы данных.
* check\_and\_update\_database – отвечает за работу с базой данных.
* check\_and\_reset\_database - отвечает за работу с базой данных.
* update\_user\_progress - обновляет прогресс пользователя в таблице users, а также записывает результаты лучшего прохождения в таблицу пройденного уровня.
* check\_level\_completion – проверяет, проходил ли этот уровень пользователь.
* get\_level\_info – предоставляет информацию о лучшем прохождении уровня.