# **Игра «Коробочки»**

## **Пояснительная записка**

### **Паспорт проекта:**

Автор – Шубарин Кирилл Владимирович

Используемые технические средства: Python, PostgreSQL (SQLite)

Используемые фреймворки: pygame, pygame\_gui

### **Краткое описание:**

Проект представляет собой 2-D игру, воссоздающую одну из первых мобильных игр «Boxes». По своему сюжету – это тетрис, в котором все элементы – это кубики, летящие с неба, но двигать их может только главный персонаж, находящийся на земле. При этом если на кубике есть ещё один кубик, то нижний не может быть передвинут, равно как и два и более кубика лежащие рядом друг с другом. У игрока в арсенале есть прыжок, позволяющий прыгать на один кубик вверх, набор жизней, которые отнимаются при попадании падающим кубиком по игроку сверху, и возможность толкать коробки в двух направлениях (влево и вправо).

При собранной линии, все коробки входящие в её состав – уничтожаются, а игрок получает очки в размере количества собранных коробок. При наборе определенного количества очков, уровень сложности игры повышается, благодаря увеличению скорости падения коробок. Игрок проигрывает, если у него не осталось ни одной жизни, или если в одном из столбцов уровень коробок достиг «потолка»

### **Архитектурное устройтсво:**

Вся игра создается в едином классе-диспетчере Game. Он следит за состоянием игрового поля, позволяет управлять музыкой, ставить игру на паузу, координирует отрисовку объектов и их взаимодействие.

Объекты, использующиеся классом Game:

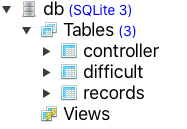
1. Player – класс главного героя. Выполняет отрисовку анимации спрайта, управляет перемещением главного героя (не учитывая движение коробочек)
2. Tile – класс отвечающий за взаимодействие коробочек друг с другом и главным героем. Каждая коробочка «знает» направление доступное для перемещения.
3. GameOver – класс финального экрана, отвечает за окончание игры, отрисовку статистики и управления приложением
4. Particle – класс анимации попадания коробкой в игрока. Создает эмиттер маленьких дракончиков, разлетающихся в разные стороны при попадании коробкой в главного героя
5. StatusHearts, StatusLevel, StatusScore – классы для настройки отображения игровой информации в строке состояния

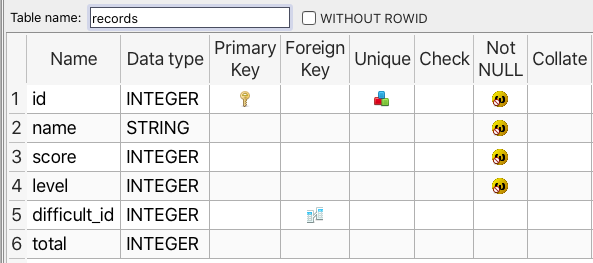
Все классы имеют доступ к функциям:

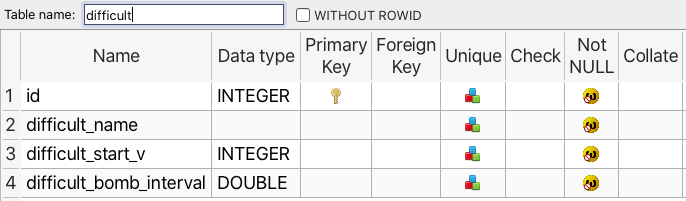
1. load\_image – загрузка изображения, и преобразование его в объект pygame.image (умеет вырезать картинку на хромокее)
2. terminate – завершение работы приложения
3. create\_particles – генератор частиц. Управляет созданием эмиттера частиц для анимации

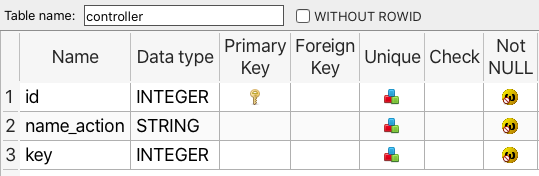
**База данных** для простоты развертывания приложения устроена на SQLite. База содержит основную таблицу *records,* вспомогательную *difficult* и таблицу с настройками управления *controller.* При помощи вспомогательной таблицы происходит конфигурация игрового процесса в зависимости от выбранного уровня сложности, а также через foreign key собирается полная информация об успехе игрока в таблице *records.*

Схема данных:









### Архитектурные особенности:

1. При смене уровня, скорость коробочек инкрементируется на единицу, благодаря методу класса Tile с использованием @classmethod. Это решение позволяет также изменить внешний вид коробочек при изменении уровня (в нашем исполнении это смена цвета коробочек
2. Нажатая клавиша движения обрабатывается отдельно для объектов типа Tile и Player, что позволяет настроить каждое движение независимо
3. Все константы вынесены в модуль constants, что позволяет очистить область видимости и освободить пространство имен, а также быстро конфигурировать игровой процесс

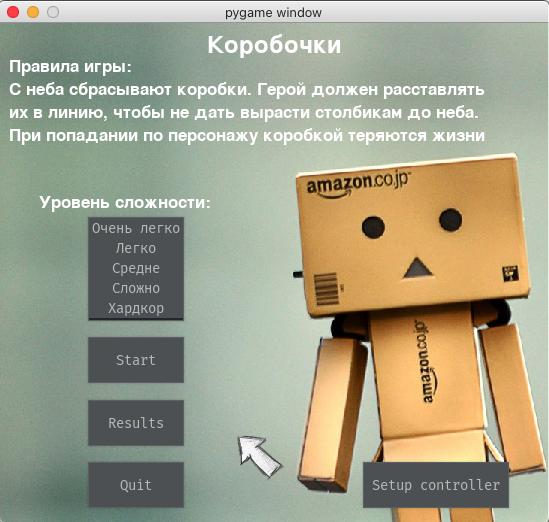
  
Рис. 1. Стартовый экран



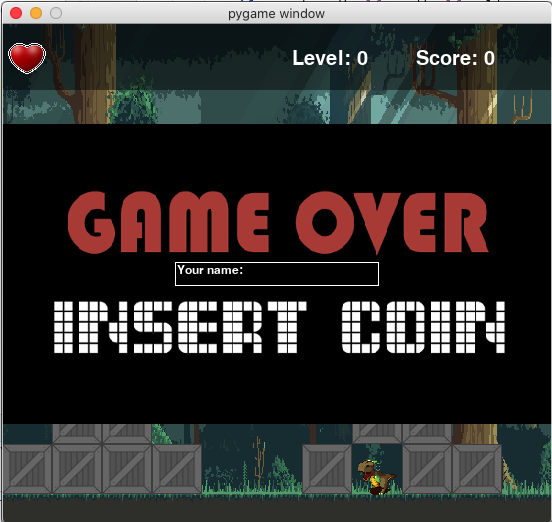
Рис. 2. Таблица рекордов



Рис. 3. Нулевой уровень  
  
Рис. 4. Меню паузы



Рис. 5. Новый уровень

  
Рис. 6. Попадание в героя  
  
Рис.7. Запись в таблицу рекордов

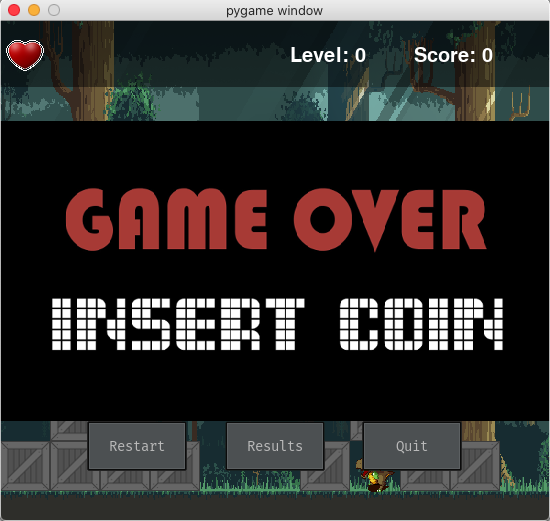


Рис. 8. Выбор действия после проигрыша