**Arkanoid+ (Арканоид плюс)**

Пояснительная записка

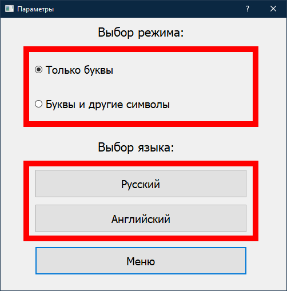
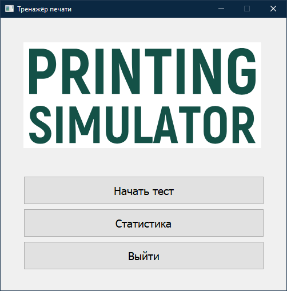
Игра Арканоид+(ArkanoidPlus) должна подарить пользователю новый опыт игры в "шарик, отскакивающий от платформы". Теперь, по мимо старых, всем известных механик, в игре появится новый элемент - заряд платформы. Чтобы перемещение платформы стало доступным, игрок должен будет периодически "заряжать" её правильными ответами на вопросы викторины, при этом он не должен забывать и о шарике, который может упасть в любой момент, вызвав проигрыш. Главная цель – наибольший счёт.

Реализованные игровые механики:

1. Отскок шарика от платформы, верхней, левой и правой границ;
2. Разрушение блоков, с которыми столкнулся шарик;
3. Ускорение темпа игры с течением времени (каждые 30 секунд);
4. Механика энергии (по истечении энергии платформа полностью останавливается);
5. Механика Викторины (правильные ответы на вопросы увеличивают количество энергии платформы, неправильные – уменьшают)
6. Система бонусов (зелёный блок увеличивает количество энергии, желтый полностью её восполняет,);
7. Система наказаний (красный блок понижает количество энергии, голубой увеличивает скорость шарика, фиолетовый меняет текущий вопрос викторины);
8. Система очков (начисляются за ломание блоков, причём за разные цвета – разное количество очков, а также за правильные ответы по викторине, за неправильные ответы очки вычитаются. В конце игры все очки умножаются на модификатор скорости, выводя итоговый счёт)

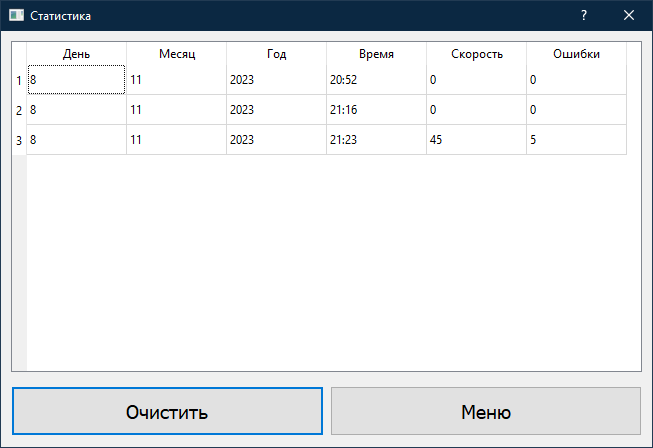
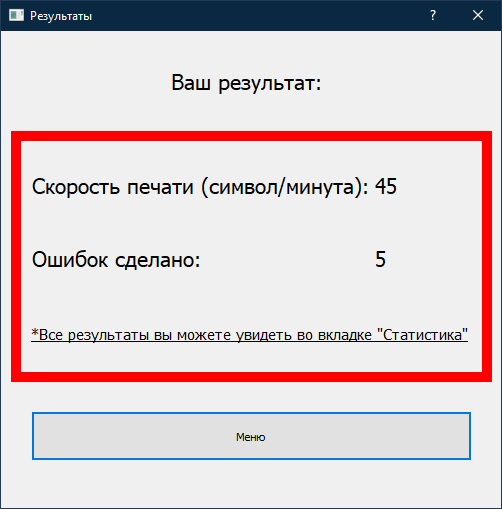
Используемые библиотеки:

* sys (для корректной работы приложения и работой с кодами возврата)
* os (для корректной работы приложения с операционной системой)
* PyGame (GUI-библиотека)
* Sqlite3 (работа с БД)
* PyGame Mixer (работа со звуками)
* Random (для составления случайного набора блоков на уровне)

Карта интерфейса: (Это шаблон моего старого проекта, добавь по аналогии по этому)



(Menu) (Rules) (TypingSpeedApp)



(Statistics) (Rezult)

Игра расположена в двух файлах кода. В main.py располагается меню игры, состоящее из (Заполни сам), а в файле game.py – непосредственно сама игра, состоящая из одного окна.

**Main.py**

Здесь ты поясняешь за свою работу

**Game.py**

Весь код игры можно поделить на 4 части:

1. Объявление констант (неизменяемых переменных) и цветов;
2. Создание классов;
3. Инициализация игры;
4. Главный цикл игры.

Константы и цвета (рис. 1)

В данной области кода сначала объявляются различные постоянные: размеры окна, игровая зона (ограничение для перемещения шарика, платформы, генерации блоков), размеры платформы, шарика, блоков, максимальное количество блоков в строке, максимальная скорость шарика, максимальное количество энергии, скорость уменьшения энергии.

Потом задаются значения цветов, использующиеся в игре: чёрный, белый, красный, зелёный, голубой, серый.

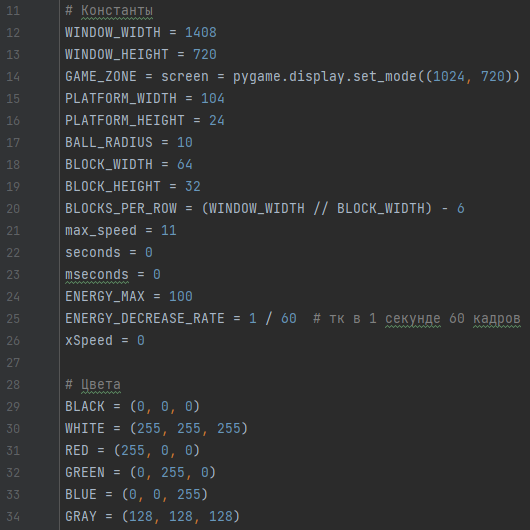


Рисунок 1. Объявление констант (неизменяемых переменных) и цветов.

Создание классов

Всего в игре используется 9 новых класса. 3 из них основные элементы игры: Platform (класс платформы), Ball (класс шарика), Block\_WHITE (класс обычного блока). Также есть 5 разновидностей обычного, белого блока: Block\_RED, Block\_GREEN, Block\_BLUE, Block\_YELLOW и Block\_VIOLET. Последний класс – Particle (класс частиц).

**Класс Platform** (рис. 2.1) рисует платформу, подгружает картинку «platform\_AKTIVE» или «platform\_NONAKTIVE» (в зависимости от количества энергии), перемещает её на текущие координаты мыши (при этом, если энергия будет равна 0, перемещение прекратиться), а также ограничивает её перемещение в пределах игровой зоны.

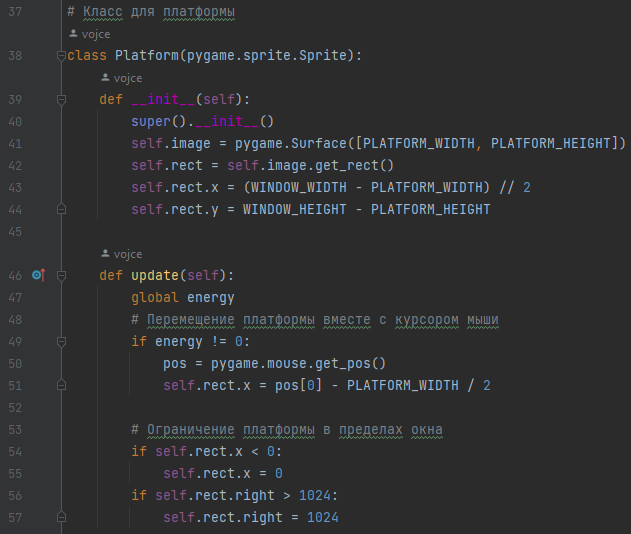


Рисунок 2.1. Класс платформы

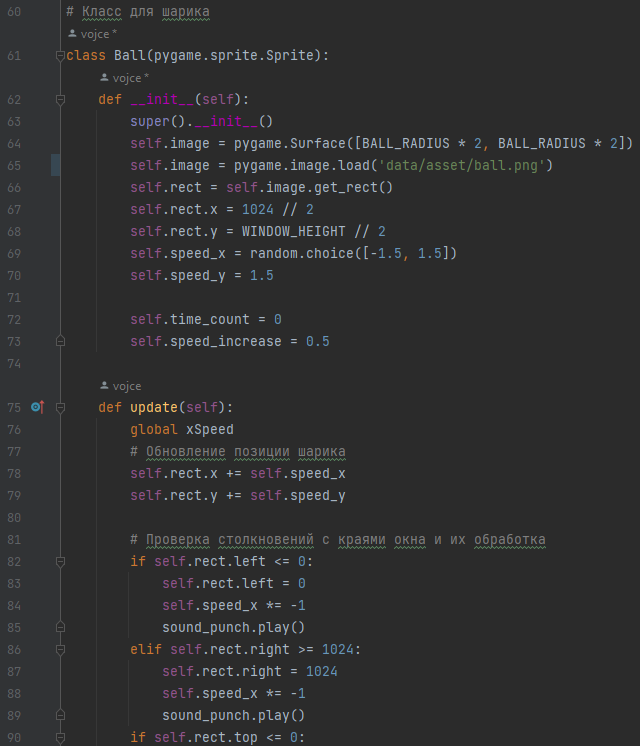
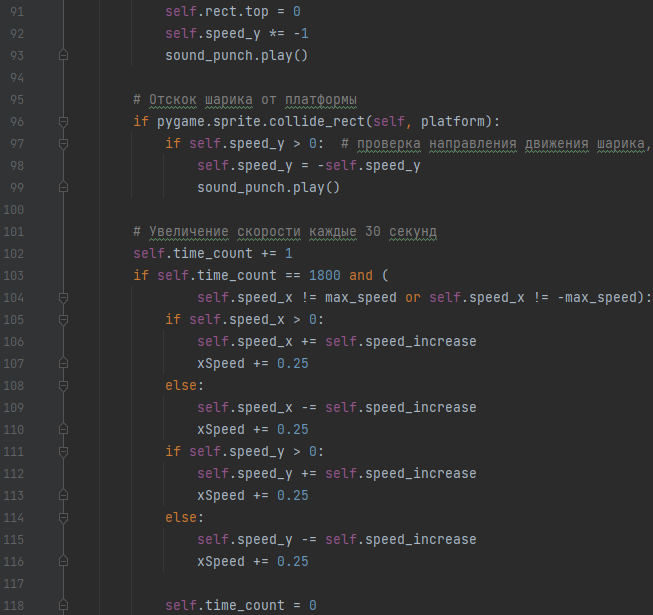
**Класс Ball** (рис. 2.2) рисует шарик, подгружает ассет «ball.png», задаёт формулу для обновления его позиции, обрабатывает столкновения с коллизиями других спрайтов, увеличивает свою скорость каждые 30 секунд.

Рисунок 2.2. Класс шарика

**Классы блоков** рисуют блоки и дают им ассет в зависимости от самого класса.

**Класс Particle** (рис. 2.3) рисует частичку, подгружает картинку «particle.png», задаёт случайную скорость по осям, уничтожает излишек, при пересечении границ игровой области.

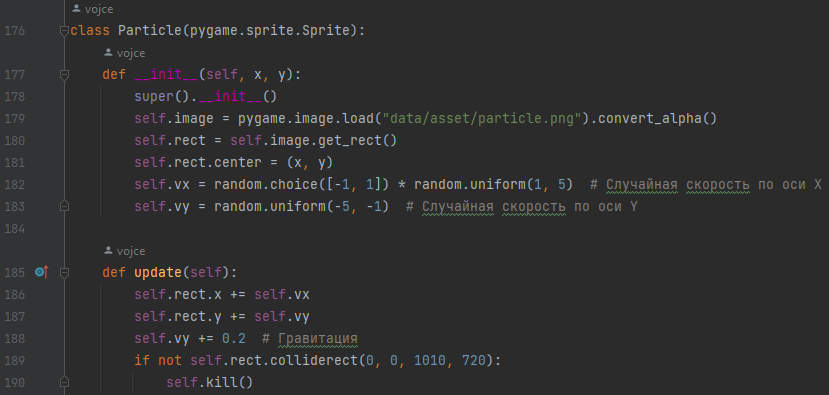


Рисунок 2.3. Класс частиц

Инициализация игры

В этой части кода происходит создание игрового окна, поделённого на две области: игровая и область для викторины и отображения важной информации (счёта, множителя скорости, таймера, шкалы энергии). Курсор мышки скрывается.

Далее загружается фоновая музыка, а также создаются константы со звуками отскока шарика, уничтожения блока, верного ответа на вопрос викторины, неверного ответа на вопрос викторины.

После создаётся задний фон игровой области. Чтобы обстановка не наскучила игроку, каждый раз задаётся случайная картинка фона.

Затем создаются спрайты шарика, платформы, а также группа спрайтов со случайной генерацией блоков (рис. 3).

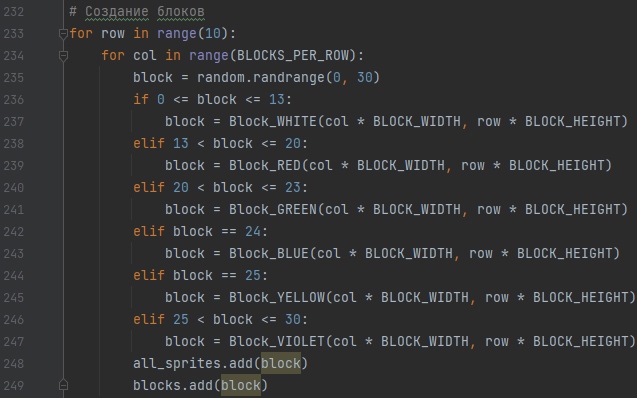


Рисунок 3. Цикл генерации блоков.

Потом рисуется полностью заполненная шкала энергии.

Ну и под конец создаётся 3 функции: функция отображения вопроса и варианта ответов в окне (display\_question), функция получения случайного вопроса из базы данных «victorina.db» (get\_random\_question), функция деления длинных вопросов на несколько строк (split\_text).

Главный цикл игры

В главном цикле игры все действия, описанные выше, получают своё развитие: обновление викторины, начисление и вычитание очков счёта за те или иные действия, обновление всех спрайтов (их положение), обновление таймера, посекундное уменьшение энергии, изменение шкалы энергии, проверка столкновения шарика с блоком (при положительном результате происходят действия, описанные в 6 и 7 пунктах раздела «Реализованные игровые механики»), проверка окончания игры (при нажатии на ESC, опускании шарика ниже игровой области), обновление экрана.

Результаты

По окончанию основного цикла создаётся новое окно, где выводится отыгранное время и итоговый счёт.