***Игра “Отражение”.***

***Пояснительная записка:***

Отражение – аркадная игра, основанная на столкновениях. В игре есть триплекс (шарик), который нужно отбивать платформой и сбивать блоки (подробнее о правилах – в информационном окне игры).

В игре есть главное окно, где предоставлено отображение рекордов, ника и фото пользователя, которое выбирает программа на основе числа открытых уровней. Кроме того, есть возможность выбора уровня и сохранения на уровне в списках с пролистыванием, нажатие на заглавие – сброс выбора. В окне настроек можно менять ник, сбрасывать данные, включать\выключать музыку и открыть окно информации.

В игровом окне есть функции паузы и сохранения. Также в нём, кроме музыки, создано звуковое оформление.

Переход между окнами осуществляется кнопками открытия окон и выходами, которые перемещают на одно окно назад в дереве окон.

**!** В сохранения передаются только очки, жизни, время и конфигурация блоков.

1. *Главный экран.*

Функционал:

1. Имеются два списка с возможностью пролистывания мышью: уровни и сохранения (на каждый уровень – свой список сохранений); **(Я создал класс виджета списка с пролистыванием и возможностью выбора элемента. Классу можно задать список элементов, на каждый – текст, который на нём отображается, и внутреннюю информацию, которая будет использоваться в коде, а также параметры кнопки справа, если она нужна.)**
2. Из них можно выбрать то, что нужно, и начать игру в новом окне. **(На каждой итерации цикла событий к объектам класса списка применяется метод process\_event(event). Он обрабатывает события сам и вызывает тот же метод у всех своих элементов. Если один из них нажат, то он меняет у списка атрибут выбранного индекса на свой индекс. Если нажат заголовок списка, выбранный элемент становится None.)**
3. Присутствует функция выхода из игры; **(Я создал класс кнопки, который так же, как и список, обрабатывает события и вызывает переданный в свой инициализатор слот при нажатии на себя.)**
4. На табло в верхнем левом углу отображаются рекорды для каждого уровня: максимальное количество набранных очков; минимальное время выигрыша; общее количество побед и поражений. **(Информация берётся из БД. В начале работы ставится количество побед и поражений, при изменении выбранного уровня класс списка вызывает переданный в него слот, обновляющий окно, в том числе рекорды, которые записаны на каждый уровень в БД.)**
5. Отображение ника и фото (картинку ставит программа в зависимости от уровня игрока). **(Ник в начале работы берётся из БД, индекс названия картинки (фото) вычисляется на основе отношения количества открытых уровней ко всем существующим.)**
6. При нажатии на кнопку настроек они открываются в новом окне. **(Использован тот же класс кнопки, что и для выхода.)**
7. *Игровое* *окно*.

Функционал:

1. Имеется экран, внизу передвигающаяся платформа. Вначале на платформе лежит шарик. При нажатии на стрелку вверх он отрывается от платформы и летит под определённым углом. **(На триплекс (шарик) и платформу созданы классы со всем необходимым. Для запуска используется метод collide\_triplex(point) (см. пункт 5).)**
2. Пока шарик на платформе, им можно управлять стрелками (Shift + влево или вправо). **(В классе шарика есть метод process\_move(), который обрабатывает событие в зависимости от родительского флага start, определяющего режим игры, позволяя управлять также и зажатием клавиши. Метод move отвечает за само передвижение и его ограничения.)**
3. Платформу можно передвигать зажатой мышью или стрелками (только горизонтально). **(В классе платформы есть метод process\_event(event), который обрабатывает событие и запоминает координату захвата. Если она не None, платформа передвигается в зависимости от MOUSEMOTION. Стрелки реализованы по тому же механизму, сто и у триплекса.)**
4. Вверху находятся стены и блоки. Шар отскакивает от них по законам оптики (под тем же углом, что и прилетел). Как только шар наталкивается на блок, тот разрушается и из него падает артефакт (не всегда). Если он упадёт на платформу, будут срабатывать свойства артефакта. **(В игре существуют классы блоков и сокровищ и бортиков. Внутри класса блока есть также классы бортиков блока, по которым определяется направление отталкивания (соответствующая скорость шарика умножается на -1). Отскоки от бортиков происходят так же. При наталкивании на блок к нему вызывается метод crush(), который отвечает за его разрушение. Кроме того, в блоках происходит случайный выбор из классов сокровищ, объект выбранного класса формируется при разрушении и дальше существует самостоятельно до падения. При столкновении с платформой срабатывает метод эффекта сокровища.)**
5. От платформы шарик должен отскакивать иначе: в зависимости от позиции, на которую прилетел (центр – 90о, края – почти горизонтально, между ними – соответственно). **(В классе платформы есть метод, задающий словарь скоростей на диапазоны координат. Он применяется при инициализации или при изменении размера платформы. Метод collide\_triplex(point) находит нужный диапазон и задаёт триплексу соответствующие скорости по осям координат.)**
6. Цель - сбить все блоки и не дать шарику упасть. При падении шарика теряется жизнь и он снова оказывается на платформе. **(На каждой итерации игрового цикла к шарику применяется метод update(), который отвечает за его перемещения и столкновения. При падении (пересечении линии смерти окна) шарик вызывает у окна метод begin\_die(), который начинает смерть платформы. Класс платформы переходит в режим анимации разрушения, в конце классы триплекса и платформы пересоздаются или происходит поражение. За выигрыш отвечает метод win(), который вызывается при отсутствии на поле блоков (не считая скандиевые).)**
7. Очки зачисляются за разбитые блоки и некоторые пойманные артефакты. Время – реальное, в минутах и секундах от первого старта. **(Метод crush() у блоков также начисляет очки за разрушения. У некоторых сокровищ начисление является эффектом. Для хранения времени использован модуль datatime, его увеличение происходит по пользовательскому событию SECOND, которое работает по таймеру.)**
8. Также в окне существует возможность поставить игру на паузу (клавишей), сохранить игру и выйти (кнопками на экране или клавишами). **(Использован тот же класс кнопки, что и в главном окне (их 2: кнопка-текст и кнопка-картинка). В их методе process\_event(event) кроме прочего присутствует обработка клавиатуры на основе переданных в инициализатор клавиши и модификатора, которые нажимают кнопку.)**
9. *Настройки:*

Функционал:

1. Возвращение в главное окно. **(Запускается главное окно, и закрываются настройки).**
2. Переход в информационное окно. **(Запускается информационное окно, и закрываются настройки).**
3. Включение и выключение музыки. **(По умолчанию музыка включена. В информационных окнах играет одна музыка, а в игровом другая. Изначально с помощью модуля pygame.mixer.music для аудио загружается музыка из папки Sounds и проигрывается на повторе. При нажатии на кнопку соответствующее поле в БД заменяется на значение (True или False). Из него же берётся атрибут, который определяет, будет играть музыка или нет.).**
4. Стирание всех данных. **(В базе данных в таблице user в столбцах victories и defeats значения становятся равными нулю. В таблице savings стирается всё. В таблице levels score, time и opened приравниваются значению NULL, а первый opened получает значение True).**
5. Изменение ника пользователя. **(Для этого я создала специальный класс InputBox он наподобие QLineEdit из PyQt5. Предел – длина виджета. Удалить один символ можно с помощью клавиши Backspace, сохранить окончательный ник можно с помощью кнопки, расположенной на окне.).**
6. *Окно информации:*

Функционал:

1. Отображение правил игры **(Я сделал класс TabWidget (наподобие класса в PyQt5), который имеет несколько экранов, но отображает только выбранный на табло сверху (выбор сделан с помощью всё того же класса кнопок внутри TabWidget). Кнопки меняют выбранный индекс, если индекс экрана совпадает с ним, то все его виджеты отображаются. В данном случае на каждый экран я добавил виджет картинки с правилами: основные правила, блоки и сокровища.)**

**Ответственность:**

Настройки сделала София Гусева, Главное, игровое и информационное окна – Ростислав Шишмарев.