**Пояснительная записка к проекту pygame: Игра «За двоих»**

**Автор:** Демьянов Антон Павлович

**Описание (Удачи;)):**

Проект – это игра, написанная на языке Python. При запуске пользователь попадает в основное меню программы(стартовое окно) или же главный экран, где имеются кнопки и текст «Правила игры» с подсказкой. Далее он по нажатию кнопок производит действия в игре. Он может начать новую игру, загрузить игру или выйти из игры (имеется в виду закрыть всю программу).

Если игрок начнет новую игру, то ему предложат ввести имя, после чего он попадет в начало игры, где ему нужно будет заходить в порталы и перемещаться на уровень выше и выше, чтобы пройти игру. В «игре» у пользователя будет показываться имя пользователя, которое он выбрал при старте, номер уровня и количество шагов, которые он сделал(движение осуществляется на стрелочки(шаги спрайтов - анимированные)).

При нажатии “ESCAPE” откроется “внутри игровое меню” в котором пользователь сможет по нажатию выйти в «главное меню», загрузить или сохранить игру а также выйти из программы.

Если пользователь прошел все 5 уровней(5 раз завел спрайты, каждого в свой, порталы), то появится Меню «Конец игры», где пользователь узнает, насколько хорош его результат, также он сможет увидеть результаты самых быстрых прохождений(все данные хранятся в базе данных sql, а также в эту базу записывается каждый новый результат прохождения пользователя).

Также, после прохождения игрок сможет выйти в главное меню или выйти из программы.

На протяжении всей игры пользователя будут сопровождать информативные сообщения, если он будет сохранять или загружать игру(если будет делать это правильно или неправильно), а также при попытке ввести слишком длинное имя.

**Реализация:**

(P.s.)В коде содержится достаточное количество комментариев, с мини-пояснениями.

В начале - создание screen – экрана, задание его ширины высоты. Создание экрана для самой игры – он немного больше стартового и финального экранов.

Задаются переменные x1, y1, x2, y2 – глобальные переменные для отслеживания передвижения спрайтов игровых персонажей.

Clock – для времени как ни странно

FPS – для частоты отрисовки экранов, ну это и есть фреймс пер секонд.

nLevel=0 – номер карты

level\_map=None -сюда вернут карту (в виде списка из точек решеток и тд)

username=’’ – имя пользователя

steps=0 – счетчик шагов

start\_move- флаг для отслеживания момента с движением только одного из двух персонажей(другой уперся в стену)

menu\_option=0 – флаг, скоро скажу зачем

toptk – создается окошко ткинтера для вызова его при создании игры – задать имя пользователя.

pygame.init() – инициализируем модули библиотеки pygame

задаем группы спрайтов:

all\_sprites – все спрайты

tiles\_group – спрайты-блоки (ландшафт)

player\_group – спрайты двух персонажей пользователя

tile\_width и heighе – ширина и высота клетки блока-спрайта

пишем функцию load\_image(передаем в нее название картинки из папки data), которая загружает картинку из папки data, лежащей рядом с файлом программы. Функция load\_image() вернет картинку, удалив у нее фон, если это необходимо (надо просто передать в нее второе значение = -1). Используется сразу после написания – создадим словарь tile\_images, где по ключу-названию объекта зададим значения, которыми будет являться результат работы функции load\_image. Также в список с разными положениями спрайтов загрузим картинки, чтобы потом, для создания анимации, брать уже загруженные сюда картинки(с них всех как раз удалим фон)

После этих “организационных моментов” запускается функция start\_screen(), которая отвечает за “MAIN MENU” или же главное меню, в котором пользователь начинает свое взаимодействие с игрой. Текст с правилами в переменной intro\_text, картинка фона, а также другие картинки и спрайты подгружаются с помощью функции load\_image().

Далее в start\_screen() задаются параметры для текста в переменную font и запускается бесконечный цикл, внутри которого будет задаваться положения текста, кнопок, текста на кнопках. После этого идет проверка на нажатия пользователя, если он нажал на крестик окна игры, то игра закроется, так же и после нажатия кнопки escape, а если он нажал на кнопку мыши внутри окна игры, то мы поднимаем флаг click. Если клик == тру и координаты его мыши (берем их в переменные mx и my) в момент щелчка - в координате кнопки, то вызовется соответствующая функция, иначе ничего не произойдет.

1)Пользователь нажал на кнопку new game:

Запускается цикл для ввода имени пользователем, реализованный через ткинтер, после прохождения цикла – указываются значения в переменные steps, отвечает за подсчет шагов персонажа, nLevel, отвечает за номер карты(уровня) и запускается функция game()(о ней позже).

2)Пользователь нажал на кнопку load game:

Запускается функция load\_game()(она используется только 2 раза и оба при нажатии кнопки load game – один раз в главном и один раз во внутри игровом меню, поэтому Опишу только один раз)

Функция подразумевает собой загрузку файла, который был создан именно в результате сохранения пользователя в ходе ИМЕННО ЭТОЙ ИГРЫ (кнопка save game во внутренне игровом меню), поэтому по сути эта функция считает файл и, если данные в нем корректны (по ним можно загрузить игру), то данные передадутся в глобальные переменные, а флаг menu\_optionполучит значение 3(значение 3 означает, что игра была загружена, это нужно во избежание рекурсии при загрузке игры из внутри игрового меню p.s. старая игра бы просто не закончилась). Далее вызывается функция game(), как раз после проверки флага menu\_option.

3)Пользователь нажал на кнопку EXIT- вызывается функция terminate() – по сути - просто завершение работы программы.

После всех этих кнопок происходит отрисовка – отображение всего заданного на screen и задается указанная частота кадров для отображения, потом все повторяется.

Функция game() – основная функция игры:

Вызывается вложенная функция generate\_level(load\_map()) и по итогу ее выполнения – мы инициализируем player1 и 2 – это спрайты (игровые персонажи)

load\_map(передаем одно значение - строку с названием карты) – считывает карту из файла.txt(должен лежать в папке data) и возвращает список вида: [[@@#...],[…##.]]

этот список передается в generate\_level(), где устанавливается тип ландшафта, в зависимости от карты(проверка выполняется по номеру карты), все изображения спрайтов заранее заготовлены и будут браться из словаря, как только выясним, где и кто/что находится на карте. Убиваем всех предыдущих спрайтов из групп спрайтов, чтобы удалить старые и добавить новые, т.к. карты постоянно разные. Проходимся по level - матрице с картой, беря координаты x и y, далее идут проверки, чтобы определить какой символ это на карте, чтобы задать соответствующий его условному обозначению спрайт: клетка стены – #, клетка “поля” по которой можно ходить – ‘.’ , стартовая позиция спрайтов игрока - @, ее нужно сделать и клеткой и игроком, чтобы, сначала все вместе отрисовать, а потом по этой клетке тоже можно было ходить, как и по любой другой, портал – буква “p” английская. После проверок создаем спрайт в соответствующих координатах, обращаясь к классам Tile – клетка и Player – игрок. Также используем локальный счетчик k, чтобы отслеживать, позиция это первого персонажа или второго. Если мы в “клетке” @, то по k проверяем, если игрок 1 и menu\_option!=3(мы не загружаем игру, а создаем новую), то в глобальные переменные x1, y1 передаем координаты игрока и new\_player1 – инициализируем спрайта из класса Player, передав в класс координаты игрового персонажа.

Player:

Выполняем стандартную инициализацию – добавляем спрайта в группы спрайты игроки и все спрайты, задаем позицию левого угла клетки игрового персонажа, картинку, start\_move\_player – для отслеживания момента, когда один из персонажей упрется в стену и target\_x и target\_y для анимации. Функции опишу позже.

Tile:

Такая же инициализация – закидываем в группу спрайтов всех и клеток, задаем картинку и левый верхний угол кнш же, откуда отрисовывать его в generate\_level по итогу уже спрайты player1 и 2 возвращаются и попадают в переменные player 1 и 2 в функции gamе, далее запускается бесконечный цикл, пока пользователь ничего не нажал, происходит отрисовка всех спрайтов(.draw) с заданными позициями и картинками “по умолчанию”(типа остальные – это для анимации) на screen, также выводится строка с параметрами Имя, Уровень и Сделано шагов. Происходит обновление дисплея и все по новой.

Если пользователь что-то нажал, то он попадает в проверку, где, как и в стартовом экране, проверяются разные нажатия. Из того, что еще не объяснил – pygame.key.get\_pressed() – берем, какую кнопку нажали, передаем в переменную ключ, далее смотрим: это стрелка вверх вниз влево или вправо, в зависимости от чего выполняем соответствующие изменения координаты положения спрайта, делаем это с помощью вызова функции update(x, y, p), куда передаем координаты спрайта и номер его – это спрайт 1 или 2, чтобы, в дальнейшем отследить момент, чтобы они не “наступали друг на друга” и не встали в одну клетку, двигаясь одновременно, если идут друг за другом.

Update():

Функция, где начинается движение(конец движения – update\_end) .

!!! я тут заметил, что часто говорю движение, я имею ввиду: изменение положения спрайта (изменение координат), а затем отрисовку этих изменений на экран!!!

Идет проверка – мы пытаемся походить в поле, в которое можно двинуться (не стена), далее, если движение возможно, то проверяем, какой это спрайт – первый или второй, если первый, то изменяем его координаты, и задаем их в заранее инициализированные переменные target\_x и y. Далее делаем rect.move, но двигаем только на половину пути, т.к. вторую пройдет функция update\_end. Далее меняем картинку на картинку движения в нужную сторону и переменную старт\_мув\_плеер = 1 чтобы не двигаться по серединам клеток. И глобальную переменную старт мув тоже меняем на 1, чтобы понимать, что движение уже было начато и отследить этот момент в функции конца движения. ТАКЖЕ мы “двигаем” и второго.

После функции update мы делаем break, чтобы не считать “лишнее действие” и не начать движение сразу в несколько сторон.

После выпадения мы выпадем с уже start\_move=1, что не позволит при след. запуске цикла попасть снова в начало движения, а закинет нас уже в конец. Но в этом цикле мы отрисуем “промежуточный ход” на пол клетки с изменением картинки спрайтов и переходим к след запуску цикла.

Попадаем в случай, когда старт мув = 1, т.е. !=0:

Ждем 0.1сек, чтобы проигралась “анимация” и вызываем функции конца движения для обоих спрайтов.

Update\_end() – проверяем, что движение завершить можно – переменная страт мув плеер ==1(движение уже начато), далее перемещаем на “оставшуюся” половину клетки и меняем картинку снова на “стандартную”, после чего зануляем переменную отслеживания начала движения.

(Получилось так, что глобальная переменная start\_move – отвечает за возможность начать завершение и не дать возможность начать еще одно, пока то не законченно, а start\_move\_player – отвечает за запрет на завершение движение конкретных спрайтов)

“Движение” завершено, теперь увеличим число шагов, выполненных пользователем на 1 и далее проверим, не наступил ли он, по окончанию движению сразу в оба портала двумя спрайтами, если нет – то просто все отобразим на экране и пойдем по новой. Если зашел, то menu\_option=1, т.к. у нас новая карта, но мы ее отрисовываем не в результате загрузки игры с компа, а в результате прохождения уровня. nLevel+=1, т.к. след уровень. Сразу все отрисуем, отобразим, потом подождем пол секунды, чтобы пользователь увидел, что его персонаж зашел в портал. Потом проверим на завершение игры: если уровень меньше 5ого, то создаем(отрисовываем(инициализируем)) новую карту и создаем + возвращаем спрайтов в спрайтов(ну уже generate\_level описывал). Если же уровень выше 5ого, то пользователь - прошел игру, вызовем функцию финального окна.

End\_screen(): работает по принципу, схожему со стартовым окном. Важные отличия - добавим (add\_result()) результат пользователя из таблицы sQL и поздравим пользователя, сообщив ему его результат. Также отобразим результаты быстрейших прохождений игры. На конечном экране всего 2 кнопки – в главное меню(“MAIN MENU”) и “EXIT” – выход, с выходом ясно, а вот чтобы выйти в главное меню и не попасть в рекурсию(вызвать главное меню после игры, не выйдя из игры до этого - нельзя), значит зададим значение menu\_option на 2 и после завершим функцию return, попадем сразу на проверку этой меню опции, и если игра была завершена и пользователь пытается выйти в главное меню, то снова return.

Add\_result() – работает с базой данных sql, занося туда данные пользователя или заменяя их на новые, если такой пользователь там уже есть(по имени пользователя определяет – типа разное имя – разная личность, сегодня я-человек паук и думаю и действую(играю) так, а потом, я уже супермэн и думаю по другому, так что если пользователь в локальной игре хочет побыть “разным” в плане мышления, зависящего от имени, то он может это делать)-это я типо пояснил, почему решил так заменять. Также, после добавления данных о новом прохождении – сортирует таблицу “by Steps ASC” – по возрастанию и берет результаты самых быстрых прохождений(топ 10 или меньше, если не так много раз пройдена игра). Далее в m задается построчный шаблон с надписями и возвращает его. Потом, эти надписи будет удобно написать на финальном экране.

Про нажатие эскейпа “в игре”(вызовется функция menu()) – выпадет меню, в нем также можно загрузить, сохранить игру, выйти в главное меню и выйти из игры, все это при нажатии на кнопки с нужными названиями. Если во время этого меню еще раз нажать ескейп, то меню пропадет, а вы вернетесь к игровому процессу(это реализованно с помощью menu\_option переменной, давая ей значение 0, при нажатии на ескейп, а после отрисовки и нового запуска цикла игры – пользователь меню уже не увидит, т.к. меню опция будет “убрана” в 0, также эта переменная, пока не 0 – не позволит двигать персонажей). При загрузке и сохранении будут появляться информативные месседж боксы(реализованно с помощью tkinter’а). По поводу сохранения – очень удобно можно сохранять, но не удобно воспринимать “человеку из вне” данные, которые занеслись в файл с сохранением игры(но программа умная, она сама все считывает и загружает). Также отловлен момент, когда у пользователя будет больше миллиона шагов – счетсчик шагов заменится на надпись 'Сделано очень много шагов'.

**Программы и библиотеки, использованные для работы:**

Для создания баз данных использовалась программа ‘SQLite studio’

Для редактирования картинок фона и спрайтов из интернета, я использовал «Adobe Photoshop CS6»

Для создания кода на языке Python использовалась программа “PyCharm”

Для запуска проекта необходимы библиотеки: 