**Experiencia: VideoJuegos🎮👾**

**1.- El proyecto 👾**

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamenteTienen una carpeta llamada VJ, con su editor de codigo deben abrirla como projecto, una vez todo abierto en la vista de su proyecto (laterales) deberian ver algo similar a esto.

Esta distribucion de modulos y carpetas se puede ver aterrador, pero no se preocupen!   
Es solo una buena practica y nos permitira visualizar mejor los elementos que conforman nuestro juego.

Ahora a explicar un poco que es todo lo que le hemos entregado…  
**1.1- Y que es todo esto??? 😓**

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente  
Todos los modulos tienen un pequeño comentario explicando su funcion en el proyecto.

*\*modulo scenes/game.py*

Ciertos modulos importan Pygame y Random, también ciertas teclas las que importamos directamente para más fácil acceso.

Texto

Descripción generada automáticamente.

Las variables para utilizar ya están presentes y son de tipo , ustedes van a tener que definirlas correctamente.



Para la mecánica del juego van a tener que modelar dos clases y sus dos métodos init y update.

Texto

Descripción generada automáticamenteTexto

Descripción generada automáticamenteUpdate va a actualizar los elementos del juego cada frame, su posición, velocidad ….

* jorge.py
* bug.py

Para hacer un projecto con Pygame hay que iniciar sus módulos, esto lo hacemos con pygame.init()

Imagen de la pantalla de un celular con letras

Descripción generada automáticamente con confianza baja

* game.py

Qué es un juego sin una pantalla? Nada.

Como estamos jugando con interfaces gráficas, vamos a indicarle al programa las características de nuestra pantalla que genere un display y que imagen va a mostrar la pantalla.

Todo el siguiente código está en game.py

Texto

Descripción generada automáticamente

* game.py

Como mencionamos en la presentación, los juegos cuentan con un reloj interno, clock va a ser nuestro reloj.

Texto

Descripción generada automáticamente

Como vamos hacer que aparezcan enemigos??? Bueno vamos a crear un evento ADDENEMY que se encargara de eso.

Imagen de la pantalla de un celular con letras

Descripción generada automáticamente con confianza media

No nos debemos olvidar de crear la instancia de nuestro jugador, también por conveniencia vamos a crear contenedores de sprites, enemies y all\_sprites que nos permitirán actualizarlos mas rápidamente.

Texto

Descripción generada automáticamente

¡Una vez que tengamos listo todo lo anterior y completemos el gameloop vamos a tener listo nuestro primer videojuego!



**2.0- El código**

Ahora que entienden el código que les entregamos podemos comenzar a codificar el juego.

Podemos asegurarnos que Pygame esta bien instalado ejecutando el siguiente código por la CMD(consola de comandos)

****

En esta sección usaremos el módulo game.py

**2.1 – El display parte 1**

Vamos a crear la pantalla donde va a ocurrir el juego, vamos a llamar a display que pertenece a Pygame, y que tiene el método set\_mode que recibe una tupla con el tamaño de la pantalla.



Para eso le asignaremos valores a las variables SCREEN\_WIDTH y SCREEN\_HEIGHT según las medidas de nuestro fondo 1000x700.

Imagen de la pantalla de un celular con letras

Descripción generada automáticamente con confianza media

Y definiremos nuestra screen, usando el código que vimos más arriba.



Si corres el código ahora, se abrirá una pantalla en negro durante unos instantes.

**2.2.- El gameplay**

Y que juego hacemos solo con la pantalla?

Ninguno 😛

Es por eso que debemos comenzar a crear lo que le dé forma a nuestro juego, nuestro Gameloop y todos sus elementos, como mostramos en la presentación, vamos a tener a nuestro personaje (Jorge) moviéndose por toda la ventana, esquivando obstáculos (Bugs) que aparecerán de manera aleatoria a los costados de nuestra ventana. Si un Bug le alcanza a dar a Jorge el juego termina.

Y bueno como traducimos esto al código?

Vamos a tener dos clases, la de **Player** y la de **Enemy que.**

**-Player va a poder moverse mientras este dentro de la ventana.**

**-Enemy aparecerá cada cierto tiempo en el borde de nuestra ventana.**

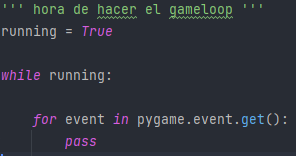
**-Si Player impacta Enemy, se cerrará la venta**

**2.3.- Game loop parte 1**

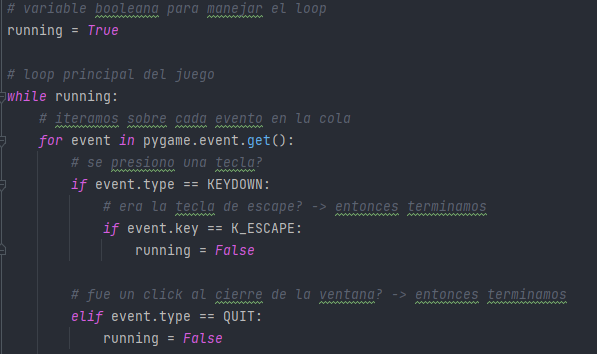
Para poder realizar el gameplay de nuestro juego vamos a tener que generar un loop, un game loop, que nos permitirá registrar todos los eventos que pasen. ¿Qué eventos tenemos ya? Los de registro de input de usuario, específicamente usaremos que registra cuando una tecla esta presionada y  que registra cuando apretamos el botón de cierre de ventana.

Todos los eventos que se registren entre las iteraciones (repeticiones) del game loop, quedan registradas en , esto nos permitirá iterar sobre cada evento que haya ocurrido.

Entonces nuestro game loop debería verse como el siguiente ejemplo:



Los eventos guardan información que se puede acceder, como por ejemplo que tipo de evento ocurrió. Con esto podemos saber que tecla fue apretada por ejemplo, si actualizamos el loop anterior para que registre los eventos que mencionamos, obtenemos algo como.



Ahora nuestra ventana seguirá existiendo hasta que apretemos el ESCAPE o cerremos la ventana.

**2.4 – El display parte 2**

Recuerdan que los programadores comienzan contando desde el 0? Pues los display también. La ventana que creamos tiene las siguientes posiciones, como una matriz, que nos permitirá poner objetos sobre ella dando las coordenadas correspondientes.

Imagen que contiene Forma

Descripción generada automáticamente

Bueno ahora que masterizamos nuestra poderosa ventana en negro, es hora de darle un toque más colorido, vamos a agregar una imagen de fondo.

Cual es un buen lugar para que Jorge esquive los BUGS? Pues dentro de un computador! Para el fondo del juego van a usar la imagen que les dimos .

Y como es que cargamos una imagen en Pygame? Pues…



* Scenes/game.py

Que es todo ese string dentro del método **load()** ? pues es la ruta de la imagen…  
el camino que debe seguir el código para encontrar el archivo, ingresamos a la carpeta /assets y encontramos la foto que queremos.

Que hace ? Escalara la imagen para que sea proporcional con el tamaño de la ventana… En este caso no hará nada ya que la ventana que creamos es del mismo tamaño que la imagen ya que escalando se puede perder calidad en la imagen.

Pero esto no es todo, debemos ahora “dibujar” la imagen en la ventana, para hacerlo dentro del loop escribiremos:

Texto

Descripción generada automáticamente

Que hace ? Toma dos argumentos, el QUE se dibujara y el DONDE en la superficie que escojamos, en este caso screen.

Ahora si ejecutamos no habrá pasado nada, porque?

Pues le pasamos la información a la ventana pero no la volvimos a dibujar, hay que actualizarla en el loop para que la pantalla refleje todos los cambios que ocurrieron durante el Game loop.

Entonces al final del game loop hay que escribir.



Que hace ? Actualiza la ventana con todo lo que le hemos dibujado usando blit().

Y listo ya hemos puesto nuestra primera ventana con fondo!

Imagen de la pantalla de un computador

Descripción generada automáticamente con confianza baja

**2.5 – Jorge y los BUGS**

Nuevamente tenemos el problema, es entretenido como juego lo que llevamos? Aun no, hay que añadirle a Jorge y los BUGS.

Texto

Descripción generada automáticamenteTexto

Descripción generada automáticamente

* jorge.py
* bug.py

“Ambas clases tienen algo raro no? Pues es que heredan, ¿Que es heredar? Aun no lo nescesitais”

-Jorge (posiblemente)

Lo que les interesa y les sirve… Va a permitir relacionar nuestro código con un Sprite. Pero hay que definir que vamos a usar como Sprite, en su código encontraran cuatro variables.

Imagen de la pantalla de un celular con letras

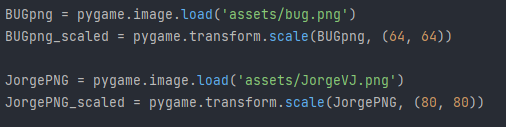
Descripción generada automáticamente con confianza mediaTexto

Descripción generada automáticamente

* jorge.py
* bug.py

Al igual como lo hicimos con el fondo, usaremos pygame.image.load(). Con las variables BUGpng y JorgePNG.

Para que son las scaled? Si queremos hacer los enemigos o a Jorge más grandes, más pequeños podremos escalar la imagen que tenemos.



Que hace ? Toma dos parámetros la imagen de referencia y su nuevo tamaño y entrega la imagen corregida.

Texto

Descripción generada automáticamenteAhora podemos empezar a trabajar con las clases, empezando con Player.

* jorge.py

Que hace todo eso que pusiste? Pues …

Pygame utiliza “Surfaces” para dibujar, Piénselo como una hoja vacía en la que podemos poner cualquier color usando RGB (Red Green Blue) o poner sprites e imágenes.

Ponemos el Sprite en el surf de la clase, la siguiente línea nos permite que la imagen mantenga sus colores y rect nos dejara dibujarlo en la pantalla.

Y ahora vemos el Enemy

Texto

Descripción generada automáticamente

* bug.py

Y el rect?? Pues…

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

* bug.py

Lo que estamos haciendo es darle una posición inicial y una velocidad aleatoria, partirán 100 pixeles a la derecha de la pantalla (no podremos verlos al principio) , a una altura que desconocemos y a una velocidad de 3 a 5 pixeles por frame.

Y bueno también queremos que se muevan, no? Para eso actualizaremos el método update() .

En Player, vamos a entregarle pressed\_keys que tiene va a tener la información de las teclas presionadas, vamos a usar las flechas del teclado (se importaron!) usando move\_ip moveremos el personaje en los ejes X e Y, también revisaremos que no se pueda salir de la pantalla.

Pantalla de computadora con letras

Descripción generada automáticamente con confianza media

* jorge.py

Ahora revisaremos el update de Enemy.

Vamos a usar la speed que definimos antes, moveremos los BUGS a través de la pantalla, y si salen de ella se destruirán.

Texto

Descripción generada automáticamente

* bug.py

Y si corremos el código…. NADA PASA?????

**2.6 – Jorge y los Bugs parte 1.5**

Bueno rect nos dejara dibujar todo lo que creamos en pantalla … pero el no va a hacerlo, hay que usar blit al igual que con el fondo….

Hay que usar blit con cada enemigo???

Que pasa si hacemos 100???

Crearemos una instancia de player:

Texto

Descripción generada automáticamente

* game.py

Pygame tiene la solución, usando sprite.Group() , vamos a agrupar todos nuestros objetos para que sea más fácil dibujarlos todos en la pantalla.

Aprovecharemos de crear dos, uno para todos los objetos y otro para solo los enemigos.

Texto

Descripción generada automáticamentePorque uno solo para enemigos? Pues la usaremos más adelante tengan fe.

* game.py

Y agregamos al game loop, antes del pygame.display.flip(), los siguiente:

Texto

Descripción generada automáticamente

* game.py

Y si corremos el código…aparece Jorge y ……

De nuevo no pasa nada?????

Hay que añadir los update al game loop.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

* game.py

Listo ahora podemos ver y mover a Jorge!!!!

**2.7 – Generando BUGS**

Habrán notado que no han aparecido los bugs, es porque no hemos creado ninguna instancia de ellos así que es hora de crear un evento que nos genere bugs.

Para hacer un evento propio hay que saber que internamente Pygame tiene ordenado los eventos como números (INT), Tiene un máximo de 32 eventos pero Pygame ya usa 23, tenemos del 24 al 32 para usar como queramos, pygame.USEREVENT es el 24 y el que vamos a usar es el 25.

Texto

Descripción generada automáticamente

* game.py

Que es lo que hace ? Colocara el evento ADDENEMY a la cola de eventos cada 600 ticks.

Pero ahora hay que crear las instancias de BUG en el game loop.

Texto

Descripción generada automáticamente

* game.py

Detectamos el tipo de evento, añadimos la nueva instancia de Enemy a los groups y listo!

**2.8 – El tiempo**

Todo pasaba muy rápido, no? Es que los ticks van a lo máximo que pueden, mientras mejor procesador más rápido el juego, Para regularlo usaremos Clock del modulo time de Pygame.

Texto

Descripción generada automáticamente

* game.py

Y en el game loop vamos a indicar a cuantos frames por segundo se va actualizar la ventana y los elementos del juego, añadiremos el código al final del game loop.



Si ahora corremos el juego ira a la velocidad que le pusimos!!!

**2.9 – Esquivando BUGS**

Vamos a tener que calcular el recuadro de jugador y lo de los bugs y ver si se tocan todo el rato? No

La magia de Pygame es que vamos a usar una función que nos entrega para ver si Jorge ha chocado con los BUGS.

Texto

Descripción generada automáticamente

**3.0 – JUGAR Y MEJORAR**

**Imagen de la pantalla de un video juego

Descripción generada automáticamente con confianza media**

**AHORA ya has programado tu primer juego! Aun tienes tiempo? Trata de mejorarlo y añadirle funciones para personalizarlo!!!**