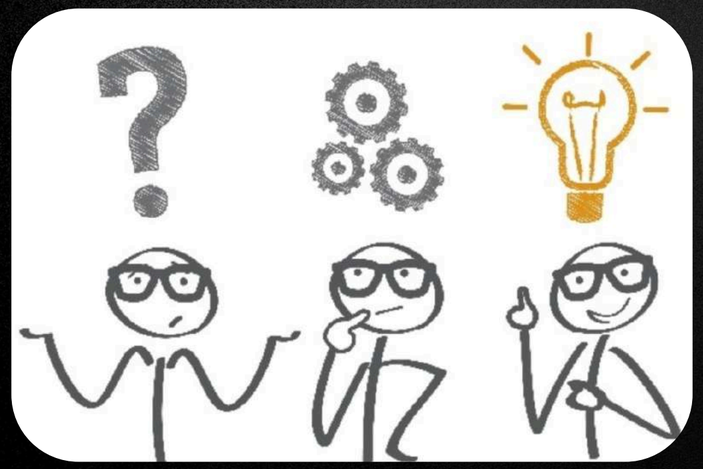
VIDEOJUEGO INTERACTIVO CON PYGAME

Objetivo:

El propósito de este proyecto es desarrollar un videojuego interactivo que permita aplicar los conocimientos aprendidos a través de la creación del videojuego, se busca comprender mejor el funcionamiento de la programación dentro de un proyecto real y cómo puede utilizarse para resolver problemas de manera práctica.

Empezar



¿QUÉ PROBLEMÁTICA RESOLVEMOS?

Solución

Como solución a la problemática planteada es que el estudiante con dificultades en su aprendizaje implemente sus conocimientos de programación y brinde mejoras a la lógica del algoritmo y del código dentro del desarrolló práctico de un videojuego interactivo mientras el aprendizaje es más dinámico y entretenido. De esta manera puede fallar, corregir errores, bugs y entender a profundidad cada paso realizado y línea de código escrita mejorando la comprensión en proyectos de gran magnitud en el mundo real laboral.

HERRAMIENTAS UTILIZADAS

Python

Python es un lenguaje de programación de alto nivel, interpretado y de propósito general para la programación de principiantes y expertos.

PYGAME

Pygame es una de las bibliotecas más populares y antiguas de Python, diseñada específicamente para el desarrollo de videojuegos en 2D.

Para este proyecto se hizo uso como lenguaje de programación a Python junto con su librería PYGAME la cual nos permitirá crear cada característica del video juego. El editor de código con el cual ejecutamos el programa es Visual Studio Code.

Durante el desarrollo del videojuego se utilizaron estructuras condicionales, ciclos, listas y funciones, así como el manejo de eventos del teclado y del mouse.

DESCRIPCIÓN DEL VIDEOJUEGO

Space Invader

Durante el juego, el jugador puede desplazarse y disparar para eliminar a los enemigos que aparecen en la pantalla. A medida que avanza el juego, la dificultad aumenta así como rapidez del mismo.
El sistema responde de manera inmediata a las acciones del usuario, actualizando el puntaje y controlando el avance de los niveles. El videojuego finaliza cuando el jugador colisiona con un enemigo o cuando un enemigo alcanza el límite inferior de la pantalla.

```
class Nave:
    def __init__(self):
        self.imagen_original = pygame.image.load("nave.png").convert_alpha()
        self.imagen = pygame.transform.scale(self.imagen_original, (50, 40))
        self.rect = self.imagen.get_rect()
        self.rect.centerx = ANCHO // 2
        self.rect.bottom = ALTO - 10
        self.velocidad = 5
        self.lasers = []
        self.cool_down = 0
```

Creación del jugador:
Se encarga de cargar la imagen de la nave, definir su tamaño, su posición inicial en la pantalla y su velocidad.

PARTES IMPORTANTES DEL CÓDIGO

Creación y movimiento jugador

Movimiento del jugador con el teclado: Permite mover la nave a la izquierda y a la derecha usando el teclado. Además, evita que la nave salga de los límites de la pantalla.

```
def mover(self, teclas):
    if teclas[pygame.K_LEFT] and self.rect.left > 0:
        self.rect.x -= self.velocidad
    if teclas[pygame.K_RIGHT] and self.rect.right < ANCHO:
        self.rect.x += self.velocidad
```

Creación, movimiento enemigo y acción de disparo

```python
def disparar(self):
    if self.cool_down == 0:
        nuevo_laser = Laser(self.rect.centerx - 2, self.rect.top, -10)
        self.lasers.append(nuevo_laser)
        self.cool_down = 20
```

Disparo del jugador:
Este código permite que la nave dispare cuando el jugador presiona la tecla correspondiente.

```python
ass Enemigo:
    def __init__(self, x, y):
        try:
            self.imagen_orig = pygame.image.load("enemigo.png").convert_alpha(
            self.imagen = pygame.transform.scale(self.imagen_orig, (40, 30))
            self.usa_imagen = True
        except:
            self.usa_imagen = False

        self.rect = pygame.Rect(x, y, 40, 30)
        self.color = (random.randint(150, 255), 50, 50)

    def mover(self, vel_y):
        self.rect.y += vel_y
```

Creación y movimiento de los enemigos:
En esta parte del proyecto se implementó la creación de los enemigos, definiendo su posición, tamaño y movimiento dentro del videojuego. Los enemigos se desplazan hacia abajo en la pantalla.

```python
if e.rect.colliderect(jugador.rect) or e.rect.bottom > ALTO:
    corriendo_juego = False
for l in jugador.lasers[:]:
    if l.rect.colliderect(e.rect):
        enemigos.remove(e)
        jugador.lasers.remove(l)
        puntos += 10
        break
```

Detección de colisiones (disparo vs enemigo):
Detecta cuando un disparo impacta a un enemigo o cuando un enemigo choca con el jugador.

Restricciones y sistema de puntaje.

```python
def obtener_max_puntaje():
    try:
        with open("puntaje.txt", "r") as f:
            return int(f.read())
    except:
        return 0

def guardar_puntaje(nuevo):
    max_actual = obtener_max_puntaje()
    if nuevo > max_actual:
        with open("puntaje.txt", "w") as f:
            f.write(str(nuevo))
```

Puntajes guardado en archivo:
Guarda el puntaje más alto del jugador en un archivo de texto.

CONCLUSIÓN

El desarrollo del videojuego permitió aplicar de forma práctica los fundamentos de programación en Python mediante Pygame, convirtiendo la teoría en una solución funcional. La implementación de mecánicas, puntuación y manejo de eventos fortaleció la lógica de programación, la resolución de problemas y la organización del código. Además, el proyecto evidenció la importancia del trabajo colaborativo y la correcta distribución de tareas para alcanzar objetivos complejos dentro de plazos establecidos.

"Si compila a la primera, desconfía; si no compila, persiste."

Terminamos

MUCHAS GRACIAS