Programación

Actividad Práctica Pygame

Enunciado:

Space Invaders (Invasores del Espacio) es un videojuego de Arcade diseñado por Toshihiro Nishikado y lanzado al mercado en 1978. El jugador deberá eliminar oleadas de alienígenas con un cañón láser y obtener la mayor cantidad de puntos posible. Para el diseño del juego, Nishikado se inspiró en Breakout, La guerra de los mundos y Star Wars. Aunque es un juego simple para los estándares actuales, fue uno de los precursores de los videojuegos modernos y ayudó a expandir la industria del sector, desde una mera novedad a una industria global.

Realice su propio Space Invaders, desarrollando en un principio; el movimiento lateral del cañón, la mecánica de disparo e impacto contra los marcianos y de ser posible un movimiento lateral del conjunto de marcianos (No es necesario que los marcianos vayan bajando ni tampoco son necesarias las defensas)**.** Puede dibujar con primitivas de pygame o utilizar sprites del juego por ejemplo del siguiente link:

https://www.spriters-resource.com/arcade/spaceinv/

El desarrollo de esta actividad es una buena oportunidad para investigar las clases pygame.sprite.**Sprite** y pygame.sprite.**Group.**

	pygame.sprite.Sprite	pygame.sprite.Group
¿Qué es?	Es una clase base para cualquier objeto del juego que tenga: • Una pygame.Surface (self.image) • Una posición o rectángulo (self.rect)	Es una colección de sprites. Sirve para agrupar y manejar múltiples objetos a la vez.
¿Ejemplos?	• Jugador, enemigo, proyectil, etc.	• Grupos de enemigos, como los marcianos de Space Invaders.
¿Ventajas?	• Permite utilizar funciones automáticas como update() y draw(). • Facilita la detección de colisiones. • Es compatible con los grupos de sprites (Group)	• Se puede invocar update() y draw() a todos los sprites del grupo en una sola línea de código. • Permite detectar colisiones fácilmente. • Organiza mejor el videojuego y genera un código mas limpio.

En este ejemplo cada enemigo será un cuadrado ROJO de 40 píxeles de ancho y 40 píxeles de largo, posicionado donde x e y lo determinen.

```
class Enemy(pygame.sprite.Sprite):
    def __init__(self, x, y):
        super().__init__()
        self.image = pygame.Surface((40, 40))
        self.image.fill("red")
        self.rect = self.image.get_rect(topleft=(x, y))
```

Ahora creamos un grupo para los disparos y otro para los enemigos, en el ejemplo automáticamente agregamos dos enemigos en distintas posiciones, estos dos enemigos forman parte del mismo grupo.

```
bullets = pygame.sprite.Group()
enemies = pygame.sprite.Group()

enemies.add(Enemy(50, 50))
enemies.add(Enemy(100, 100))
```

En este ejemplo al presionar la barra espaciadora creamos un disparo y lo agregamos al grupo de disparos.

```
# Input jugador
keys = pygame.key.get_pressed()
player.update(keys)

if keys[pygame.K_SPACE]:
    bullet = Bullet(player.rect.centerx, player.rect.top)
    bullets.add(bullet)
```

Las funciones de update actualizan la posición en pantalla, sean balas, enemigos o el jugador, deben ser implementadas por nosotros en cada clase, al llamar a la función update del grupo automáticamente se llama a la función update de cada clase.

```
# Actualizar grupos
bullets.update()
enemies.update()

# Colisiones
for bullet in bullets:
    hits = pygame.sprite.spritecollide(bullet, enemies, True)
    if hits:
        bullet.kill()
```

Al detectar que una bala del grupo de balas colisiona con algún enemigo del grupo de enemigos, se elimina la bala. También observen el True de la función spritecollide, ese True indica si el enemigo que colisiona con la bala debe ser eliminado del grupo de enemigos.

Por último dibujamos todo en pantalla. Al llamar a draw del grupo, automáticamente se dibuja cada elemento del grupo. De todas maneras podemos implementar draw en nuestras clases si queremos que el dibujado sea personalizado.

```
# Dibujar
player.draw(screen)
bullets.draw(screen)
enemies.draw(screen)

pygame.display.flip()
```

A continuación se da una recomendación en el orden del desarrollo, pero no es obligatorio seguir dichos pasos. Mas allá de la forma o el orden en que desarrollen, como recomendación; siempre planifiquen pequeños incrementos y ejecuten el juego para ver si todo funciona correctamente hasta ese momento o hay que corregir algún error ya sea ejecución o de lógica.

Incremento 1:

- Mostrar y mover el jugador
  - Pasos recomendados:
    - Crear una clase **Player o Jugador** que pueda dibujarse y desplazarse lateralmente con las teclas controlando que no se salga de los bordes.

Incremento 2:

- Mostrar un enemigo que pueda moverse de manera autónoma.
  - Pasos recomendados:
    - Crear una clase **Enemy o Enemigo** que pueda dibujarse y desplazarse de manera autónoma de izquierda a derecha y viceversa controlando cuando cambiar de sentido.

Incremento 3:

- Crear un grupo de enemigos (puede ser una fila o varias).
  - Pasos recomendados:
    - Crear un grupo para los enemigos.
    - Crear varios enemigos y agregarlos al grupo.
    - Verificar que se muevan bien en grupo (que no se solapen, etc)

Incremento 4:

- Generar disparos desde el jugador.
  - Pasos recomendados:
    - Generar la clase **Bullet** o **Disparo**.
    - Crear un grupo para los disparos.
    - Al detectar que se presionó la tecla asignada para el disparo, generar un disparo y agregarlo al grupo de los disparos.

Incremento 5:

- Detectar las colisiones entre las balas y los enemigos.