VIDEOJUEGOS
INTRODUCCIÓN A PYTHON

CLASE 09

Contenido

“

Este espacio curricular permite fortalecer el espíritu crítico y la actitud creativa del futuro graduado. Lo alienta a integrar los conocimientos previos, recreando la práctica en el campo real, al desarrollar un proyecto propio.

”

PROGRAMACIÓN I

¡Bienvenidos al curso de programación de videojuegos con Pygame! En este apasionante viaje, exploraremos las maravillas de la creación de juegos, desde los fundamentos hasta la construcción de emocionantes experiencias interactivas. A través de Python y la potencia de Pygame, descubrirán cómo dar vida a sus ideas y convertirlas en juegos reales. Aprenderemos a manejar gráficos, sonidos, colisiones y movimiento, así como a implementar mecánicas de juego emocionantes. Prepárense para desatar su creatividad, aprender conceptos clave y sumergirse en el apasionante mundo de la programación de videojuegos. ¡Únanse a nosotros y comencemos a construir mundos virtuales llenos de diversión y desafíos!

Introducción

Pygame es una biblioteca de código abierto y gratuita para el desarrollo de videojuegos en el lenguaje de programación Python. Proporciona una amplia gama de funcionalidades y herramientas que facilitan la creación de juegos 2D. Pygame se basa en la biblioteca SDL (Simple DirectMedia Layer) y ofrece una interfaz sencilla y fácil de usar para trabajar con gráficos, sonidos, colisiones, animaciones y entrada del usuario. Permite a los desarrolladores enfocarse en la lógica del juego sin tener que preocuparse por los detalles de bajo nivel. Con Pygame, es posible crear juegos desde simples hasta complejos, y es una excelente opción para aquellos que desean aprender a programar videojuegos o crear prototipos rápidos.

Cómo instalar Pygame

Para instalar Pygame, puedes seguir los siguientes pasos:

1. Asegúrate de tener Python instalado en tu sistema. Pygame es compatible con Python 3.x, así que asegúrate de tener una versión actualizada.

2. Abre una terminal o línea de comandos en tu sistema operativo.

3. Ejecuta el siguiente comando para instalar Pygame utilizando pip, que es el administrador de paquetes de Python:

```
pip install pygame
```

Si estás utilizando una versión específica de Python, asegúrate de usar el comando pip correspondiente a esa versión (por ejemplo, `pip3` para Python 3).

4. Espera a que la instalación se complete. Pip descargará e instalará automáticamente Pygame y todas sus dependencias.

Una vez que la instalación haya finalizado con éxito, podrás comenzar a utilizar Pygame en tus proyectos de programación de videojuegos con Python. Recuerda importar el módulo de Pygame en tus scripts para utilizar sus funciones y características.

Importar la librería

Para importar la biblioteca Pygame en tu script de Python, simplemente agrega la siguiente línea de código al principio de tu archivo:

```
import pygame
```

Al importar Pygame de esta manera, tendrás acceso a todas las funciones y clases proporcionadas por la biblioteca. Asegúrate de haber instalado previamente Pygame siguiendo los pasos de instalación adecuados para tu sistema operativo. Luego, puedes utilizar las funciones y clases de Pygame en tu código para crear y desarrollar tus juegos.

Interfaz GUI o CLI

La GUI (Interfaz Gráfica de Usuario, por sus siglas en inglés) y la CLI (Interfaz de Línea de Comandos, por sus siglas en inglés) son dos tipos diferentes de interfaces que permiten la interacción entre los usuarios y las aplicaciones.

La GUI es una interfaz visual que utiliza elementos gráficos como botones, ventanas, menús desplegables y cuadros de texto para facilitar la interacción con el usuario. Proporciona una representación visual de los

datos y permite a los usuarios interactuar con la aplicación de manera intuitiva a través de acciones de clic, arrastrar y soltar, y otros gestos. Las interfaces gráficas son comunes en aplicaciones de escritorio y en dispositivos móviles, y suelen ser más visuales y amigables para los usuarios que las interfaces de línea de comandos.

Por otro lado, la CLI es una interfaz que se basa en el uso de comandos de texto escritos por el usuario para interactuar con la aplicación. Los usuarios ingresan comandos específicos y reciben respuestas o resultados en forma de texto. La CLI es más común en sistemas operativos basados en texto, como las terminales de comandos en sistemas Unix o la ventana de comandos en Windows. Aunque puede requerir un aprendizaje inicial para familiarizarse con los comandos y su sintaxis, la CLI puede ser más eficiente para realizar tareas rápidas y repetitivas una vez que se dominan los comandos relevantes.

En resumen, la GUI se centra en la interacción visual y la facilidad de uso, mientras que la CLI se basa en comandos de texto para interactuar con la aplicación. Ambos enfoques tienen sus ventajas y se utilizan en diferentes contextos según las necesidades del usuario y la aplicación.

Colores en Pygame

En Pygame, los colores se pueden representar mediante una tupla de tres valores que representan los componentes "rojo (R), verde (G) y azul (B)". Cada componente puede tener un valor en el rango de 0 a 255, donde 0 representa la ausencia del color y 255 la intensidad máxima.

Pygame también proporciona una serie de colores predefinidos para facilitar su uso. Aquí tienes algunas formas de crear colores en Pygame:

1. Utilizando valores RGB:

color = (255, 0, 0) # Crea un color rojo intenso (R=255, G=0, B=0)

2. Utilizando valores hexadecimales:

color = pygame.Color('#FF0000') # Crea un color rojo intenso

3. Utilizando nombres de colores predefinidos en Pygame:

color = pygame.Color('red') # Crea un color rojo

Puedes utilizar estos colores para dibujar formas, rellenar superficies o establecer el color de fondo en tus juegos de Pygame.

Coordenadas

En Pygame, las coordenadas se manejan utilizando un sistema de coordenadas cartesianas, donde el punto (0, 0) se encuentra en la esquina superior izquierda de la ventana. Aquí tienes algunos detalles sobre cómo se manejan las coordenadas en Pygame:

1. Coordenadas X e Y: La coordenada X representa la posición horizontal y aumenta hacia la derecha, mientras que la coordenada Y representa la posición vertical y aumenta hacia abajo.

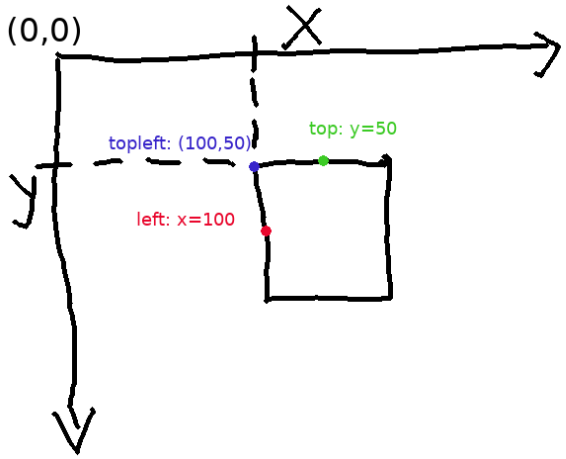
2. Tamaño de la ventana: La ventana de Pygame tiene un tamaño definido en píxeles, donde el ancho y el alto de la ventana determinan el rango de valores de las coordenadas X e Y respectivamente.

3. Dibujar en la ventana: Puedes utilizar funciones como `pygame.draw.rect()`, `pygame.draw.circle()`, etc., para dibujar formas en la ventana. Estas funciones toman las coordenadas (X, Y) como argumentos para especificar la posición de dibujo.

4. Coordenadas del ratón: Pygame proporciona la función `pygame.mouse.get_pos()` para obtener las coordenadas (X, Y) actuales del ratón en la ventana. Esto es útil para realizar acciones basadas en la posición del ratón.

5. Movimiento de objetos: Puedes actualizar las coordenadas de los objetos en el juego para moverlos por la ventana. Esto se hace modificando los valores de las coordenadas X e Y de los objetos en cada cuadro de actualización.

Recuerda que es posible ajustar el sistema de coordenadas utilizando transformaciones, como traslaciones o escalas, para adaptarlo a las necesidades específicas de tu juego.



Recuerda que las coordenadas 0,0 están situadas en la esquina superior izquierda.

Método fill

El método `fill()` en Pygame se utiliza para llenar una superficie con un color específico. Aquí tienes ejemplos de cómo utilizar el método `fill()`:

1. Llenar la ventana con un color sólido:

```
import pygame
pygame.init()
width = 800
height = 600
window = pygame.display.set_mode((width, height))
color = (255, 0, 0)  # Color rojo
# Bucle principal del juego
running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
    # Rellenar la ventana con el color rojo
    window.fill(color)
    pygame.display.update()
pygame.quit()
```

2. Llenar una superficie rectangular con un color:

```python
import pygame

pygame.init()
width = 800
height = 600
window = pygame.display.set_mode((width, height))
color = (0, 255, 0)  # Color verde
# Crear una superficie rectangular
rect = pygame.Rect(100, 100, 200, 100)
# Bucle principal del juego
running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
    # Rellenar la superficie rectangular con el color verde
    window.fill(color, rect)
    pygame.display.update()
pygame.quit()
```

Estos ejemplos demuestran cómo utilizar el método `fill()` para llenar una ventana o una superficie rectangular con un color específico. Puedes experimentar con diferentes colores y superficies para adaptarlo a tus necesidades.

Método flip

En Pygame, `pygame.display.flip()` es un método que se utiliza para actualizar la pantalla. Cuando se llama a este método, se muestran en la ventana todos los cambios realizados en los gráficos desde la última llamada a `flip()`. Básicamente, es como tomar un "pantallazo" de la ventana y mostrar los cambios en ese momento.

El método `flip()` realiza las siguientes acciones:

1. Muestra los cambios realizados en la ventana: Si has dibujado algo en la ventana, como formas, imágenes o texto, esos cambios se mostrarán en la pantalla.

2. Limpia el búfer de dibujo: Pygame utiliza un búfer de dibujo para realizar las operaciones de dibujo en memoria antes de mostrarlos en la pantalla. `flip()` vacía ese búfer y muestra los cambios.

Es importante llamar a `flip()` después de realizar todas las operaciones de dibujo que desees mostrar en la pantalla. Sin esta llamada, los cambios no serán visibles para el usuario.

Aquí tienes un ejemplo de uso de `flip()` en un bucle principal de Pygame:

```python
import pygame
pygame.init()
width = 800
height = 600
window = pygame.display.set_mode((width, height))
color = (255, 0, 0)  # Color rojo
# Bucle principal del juego
running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
    # Realizar operaciones de dibujo
    window.fill(color)  # Rellenar la ventana con el color rojo
    pygame.display.flip()  # Actualizar la pantalla
pygame.quit()
```

En este ejemplo, el método `flip()` se utiliza para mostrar los cambios realizados en la ventana después de llenarla con el color rojo. Sin la llamada a `flip()`, los cambios no se mostrarían en la pantalla.

Time.Clock

En Pygame, `pygame.time.Clock()` es una clase que se utiliza para medir el tiempo transcurrido y controlar la velocidad de actualización de un juego o una animación.

Al crear una instancia de `Clock`, se crea un objeto que puede rastrear el tiempo y proporcionar un control preciso sobre la velocidad de actualización del juego. Este objeto se utiliza en combinación con el método `tick()` para limitar la cantidad de cuadros (frames) por segundo.

Aquí tienes un ejemplo de cómo usar `Clock` en un bucle principal de Pygame:

```python
import pygame

pygame.init()
width = 800
height = 600
window = pygame.display.set_mode((width, height))
color = (255, 0, 0)  # Color rojo
clock = pygame.time.Clock()  # Crear un objeto Clock
# Bucle principal del juego
running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    # Realizar operaciones de dibujo
    window.fill(color)  # Rellenar la ventana con el color rojo

    pygame.display.flip()  # Actualizar la pantalla

    clock.tick(60)  # Limitar la velocidad de actualización a 60 cuadros por segundo
pygame.quit()
```

En este ejemplo, el objeto `clock` se crea con `pygame.time.Clock()`, y luego se utiliza el método `tick(60)` para limitar la velocidad de actualización a 60 cuadros por segundo. Esto asegura que el juego se ejecute a una velocidad constante y suave, independientemente de la velocidad de procesamiento del hardware.

En resumen, `pygame.time.Clock()` se utiliza para controlar la velocidad de actualización del juego y garantizar que se ejecute a una velocidad constante y suave en diferentes sistemas.

Los Sprite

En el contexto de Pygame, un sprite es un objeto gráfico o visual que puede ser manipulado y renderizado en una ventana o pantalla. Un sprite puede representar un personaje, un objeto, un enemigo u otros elementos visuales dentro de un juego.

La clase `Sprite` en Pygame proporciona funcionalidades y características para trabajar con sprites de manera conveniente. Los sprites pueden contener imágenes, animaciones, propiedades de colisión y comportamientos específicos. Además, los sprites pueden agruparse y organizarse en grupos, lo que facilita la gestión y manipulación de múltiples elementos visuales en un juego.

Al utilizar sprites en Pygame, puedes aprovechar diversas funciones y métodos que facilitan tareas comunes, como mover, rotar, colisionar y renderizar los objetos en pantalla. Además, los sprites suelen estar asociados con rectángulos de colisión, lo que permite detectar colisiones y realizar interacciones entre ellos.

Aquí tienes un ejemplo básico de cómo se puede crear y utilizar un sprite en Pygame:

```python
import pygame

pygame.init()
width = 800
height = 600

window = pygame.display.set_mode((width, height))

class Player(pygame.sprite.Sprite):
    def __init__(self):
        super().__init__()
        self.image = pygame.Surface((50, 50))
        self.image.fill((255, 0, 0))  # Color rojo
        self.rect = self.image.get_rect()
        self.rect.center = (width // 2, height // 2)

    def update(self):
        self.rect.x += 1
        if self.rect.left > width:
            self.rect.right = 0

all_sprites = pygame.sprite.Group()
player = Player()
all_sprites.add(player)

# Bucle principal del juego
running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    all_sprites.update()

    window.fill((0, 0, 0))  # Limpia la pantalla
    all_sprites.draw(window)  # Dibuja los sprites en pantalla

    pygame.display.flip()

pygame.quit()
```

En este ejemplo, se crea una clase `Player` que hereda de `pygame.sprite.Sprite`. El jugador (`Player`) es un sprite que se representa como un rectángulo rojo en la pantalla. La clase `Player` contiene una función `update()` que mueve el sprite horizontalmente y lo hace volver al

principio de la pantalla cuando alcanza el borde derecho.

Los sprites se agregan a un grupo (`all_sprites`), que se actualiza y se dibuja en cada iteración del bucle principal del juego.

En resumen, los sprites en Pygame permiten organizar y manipular de manera conveniente los elementos visuales de un juego, y la clase `Sprite` proporciona funcionalidades y métodos útiles para trabajar con ellos.

Animaciones

En Pygame, las animaciones se pueden lograr mediante una secuencia de imágenes que se muestran en rápida sucesión. Aquí tienes un ejemplo básico de cómo crear una animación en Pygame:

```
import pygame

pygame.init()
width = 800
height = 600

window = pygame.display.set_mode((width, height))

# Cargar las imágenes de la animación
image1 = pygame.image.load('image1.png')
image2 = pygame.image.load('image2.png')
image3 = pygame.image.load('image3.png')

animation_frames = [image1, image2, image3]  # Lista de
imágenes para la animación
current_frame = 0  # Índice de la imagen actual

# Bucle principal del juego
running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    # Actualizar el índice de la imagen actual
    current_frame += 1
    if current_frame >= len(animation_frames):
        current_frame = 0

    window.fill((0, 0, 0))  # Limpiar la pantalla
    window.blit(animation_frames[current_frame], (0,
0))  # Mostrar la imagen actual

    pygame.display.flip()

pygame.quit()
```

En este ejemplo, se cargan tres imágenes (`image1.png`, `image2.png`, `image3.png`) que forman parte de la animación. Estas imágenes se almacenan en una lista llamada `animation_frames`.

En cada iteración del bucle principal, se actualiza el índice de la imagen actual (`current_frame`) para avanzar a la siguiente imagen en la secuencia. Cuando se llega al final de la lista de imágenes, se reinicia al principio.

Luego, se utiliza el método `blit()` para mostrar la imagen actual en la pantalla en las coordenadas (0, 0).

La función `pygame.display.flip()` se utiliza para actualizar la pantalla y mostrar los cambios.

Recuerda reemplazar `'image1.png'`, `'image2.png'` y `'image3.png'` con las rutas y nombres de tus propias imágenes de animación.

Este es solo un ejemplo básico para mostrar cómo se puede lograr una animación simple en Pygame. Las animaciones más complejas pueden requerir el uso de técnicas avanzadas, como spritesheet y

Viewport

El viewport (ventana gráfica) es el área visible de una escena o contenido en un entorno gráfico o de renderizado. En el contexto de los videojuegos y gráficos 2D/3D, el viewport se refiere a la porción de la pantalla o lienzo donde se muestra el contenido visual.

El viewport determina qué parte de la escena es visible para el jugador o usuario. Puede ser una ventana rectangular que cubre toda la pantalla o una sección más pequeña dentro de la pantalla. El viewport se utiliza para recortar la escena y

mostrar solo la parte relevante para la perspectiva del jugador.

En los motores de juegos y frameworks gráficos, como Pygame, el viewport se puede configurar y manipular para ajustar el área visible de la pantalla. Esto permite implementar técnicas como desplazamiento de la cámara (scrolling) para seguir al personaje del juego, zoom in/out para cambiar la escala de la escena, o dividir la pantalla en múltiples viewports para mostrar diferentes perspectivas o cámaras.

En resumen, el viewport es el área visible de una escena o contenido en un entorno gráfico, como un juego. Es utilizado para determinar qué parte de la escena se muestra al jugador en la pantalla.

Estructura de los archivos

Creamos una estructura básica para respetar en todos nuestros proyectos. Se trata de una propuesta que apunta a tener un código ordenado y prolijo,

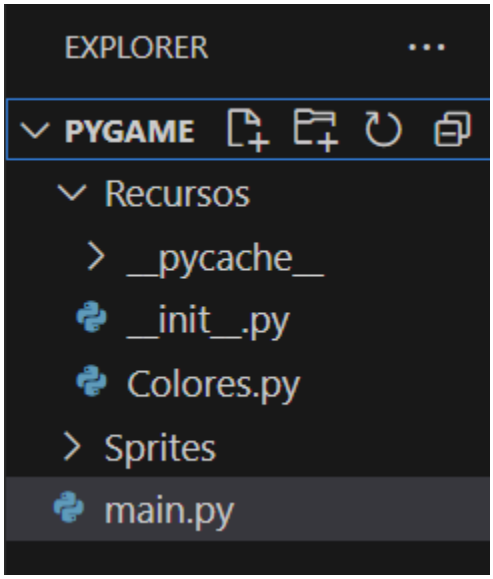
Main.py

Es El archivo principal

```python
import pygame
import os, sys, time
import Recursos.Colores as COLOR

def main():
    pygame.init()
    pantalla=pygame.display.set_mode((360,450))
    pygame.display.set_caption("Mi Primer Juego")
    reloj=pygame.time.Clock()
    fuente=pygame.font.SysFont("Arial",20)

    salir=False
    while not salir:
        for evento in pygame.event.get():
            if evento.type==pygame.QUIT:
                salir=True
        pantalla.fill(COLOR.BLANCO)

        reloj.tick(20)
        pygame.display.flip()

    pygame.quit()
    sys.exit()


if __name__=="__main__":
    main()
```

Carpeta

Creamos una carpeta para alojar todos los recursos como tuplas de colores, funciones, clases y recursos en general. Además, creamos una carpeta Sprites para contener todos los archivos de imágenes



Bibliografía

Sweigart, A. (2012). Making Games with Python & Pygame: A Guide to Programming with Graphics, Animation, and Sound. Recuperado de https://inventwithpython.com/makinggames.pdf

Pygame. (s.f.). [Página oficial de pygame]. Recuperado 2 julio, 2019, de https://www.pygame.org/news