

Como crear juegos básicos con Python utilizando la IA con ChatGPT

I. Introducción

En la actualidad, la creación de juegos y aplicaciones experimenta una revolución gracias a tecnologías innovadoras como ChatGPT, un modelo de lenguaje de inteligencia artificial desarrollado por OpenAI. Este modelo, que se integra fácilmente con lenguajes de programación como Python, permite a los desarrolladores generar contenido conversacional dinámico, llevando la interactividad y la personalización a nuevas alturas. La combinación de la programación convencional con las capacidades avanzadas de procesamiento del lenguaje natural no solo transforma los juegos en experiencias más vivas y conversacionales, sino que también democratiza el acceso a experiencias de usuario únicas y cautivadoras, rompiendo las barreras entre el usuario y la tecnología.

Esta conexión entre ChatGPT y lenguajes de programación no solo abre nuevas posibilidades creativas, sino que también destaca la importancia de adoptar estas tecnologías emergentes para impulsar la innovación en el desarrollo de software. La capacidad de crear juegos y aplicaciones más intuitivas y conversacionales redefine la interacción entre el usuario y la tecnología, sumergiendo a los usuarios en experiencias interactivas que desafían las expectativas y subrayan el papel crucial de estas nuevas tecnologías en la evolución del desarrollo de software.

II. Objetivo

El propósito de este trabajo es conocer como las nuevas tecnologías como la ingeniería de prompts ayudan a desarrollar de forma más eficiente los diferentes ámbitos de investigación. Además, sirve como guía para explorar el potencial de los modelos de lenguaje artificial desde la perspectiva de la programación.

III. Herramientas y Tecnologías Utilizadas

Como se viene mencionando este trabajo se basa en como la ingeniería de prompts puede ser útil para temas diversos pero muy específicos, por tal este trabajo se centra en el uso de ChatGPT con el fin de hacer uso de prompts. Este trabajo busca mostrar como con la buena intervención de nuevas tecnologías se pueden realizar de forma más eficientes y completas las actividades de otras disciplinas.

IV. Procedimiento

Al implementar ChatGPT para el desarrollo de este trabajo fue necesario empezar una conversación sobre el cómo se pueden crear aplicaciones con ayuda de Python como se ilustra en las primeras figuras, esto con el fin de que al IA proporcione respuestas más acertadas a preguntas futuras más específicas, además el procedimiento se dividió en diferentes partes con el fin de ir complementando la estructura del código.

Al preguntarle al ChatGPT si podía crear una aplicación en Python esta generó dos códigos diferentes en los cuales la experiencia del usuario generó más complejidad en la estructura.



ChatGPT, por favor, ayúdame a crear una aplicación en Python para un usuario con nivel de experiencia principiante. Asegúrate de seguir las convenciones de estilo PEP 8 y proporciona código con comentarios explicativos para facilitar la comprensión. Si necesitas aclaraciones sobre la funcionalidad, los requisitos o cualquier otro aspecto de la aplicación, no dudes en hacer preguntas de seguimiento. No hay límites de tiempo o longitud para las respuestas generadas



¡Claro! Estoy aquí para ayudarte a crear una aplicación en Python para principiantes. Empezaremos con una aplicación simple que permite a un usuario realizar un seguimiento de sus tareas diarias. La aplicación permitirá al usuario agregar, ver y eliminar tareas. Aquí tienes el código:



Ayúdame a crear una aplicación en Python para un usuario con nivel de experiencia avanzado. Asegúrate de seguir las convenciones de estilo PEP 8 y proporciona código con comentarios explicativos para facilitar la comprensión. Si necesitas aclaraciones sobre la funcionalidad, los requisitos o cualquier otro aspecto de la aplicación, no dudes en hacer preguntas de seguimiento. No hay límites de tiempo o longitud para las respuestas generadas



¡Claro! Vamos a crear una aplicación más avanzada en Python. En este caso, desarrollaremos una aplicación de gestión de contactos que permitirá al usuario agregar, ver, actualizar y eliminar contactos. Utilizaremos la biblioteca `sqlite3` para gestionar una base de datos SQLite que almacene la información de los contactos. Asegúrate de tener Python y SQLite instalados en tu sistema antes de ejecutar este código.



Posterior a esto se procede a preguntarle al ChatGPT si puede generar una aplicación, mas exactamente el juego de la culebrita utilizando PEP 8 que hace referencia a un conjunto de pautas y recomendaciones para escribir código en el lenguaje de programación Python. "PEP" significa "Python Enhancement Proposal" (Propuesta de Mejora de Python), y el PEP 8 es un documento que establece las directrices de estilo para el código Python.



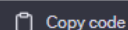
Ayúdame a crear una aplicación en Python para un usuario con nivel de experiencia avanzado para el juego de la culebrita. Asegúrate de seguir las convenciones de estilo PEP 8 y proporciona código con comentarios explicativos para facilitar la comprensión. Si necesitas aclaraciones sobre la funcionalidad, los requisitos o cualquier otro aspecto de la aplicación, no dudes en hacer preguntas de seguimiento. No hay límites de tiempo o longitud para las respuestas generadas



¡Claro! Crear una versión avanzada del juego de la culebrita en Python es un proyecto interesante. Vamos a utilizar la biblioteca Pygame para crear la interfaz gráfica del juego. Asegúrate de instalar Pygame antes de ejecutar este código:



```
bash
```



```
pip install pygame
```

ChatGPT antes de generar el código advierte que es necesario instalar Pygame, por lo cual se indaga como es el procedimiento para realizar la instalación de las librerías en Python.



¿cómo puedo realizar la instalación de librerías para Python?

Mis disculpas si no fui lo suficientemente claro. Aquí tienes una explicación más detallada:

1. El error que viste en tu sesión de Python ocurrió porque estabas dentro del intérprete interactivo de Python y escribiste el comando "pip", que no es un comando reconocido dentro del intérprete interactivo de Python. Python interpreta "pip" como un nombre que no está definido en ese contexto.
2. Para usar "pip" correctamente, debes abrir una terminal o símbolo del sistema en tu sistema operativo. Aquí hay algunos pasos que puedes seguir:
 - En Windows, abre el "Símbolo del sistema" o "Command Prompt" desde el menú Inicio o usando la combinación de teclas Win + R y luego escribiendo "cmd".
 - En sistemas basados en Unix (Linux o macOS), abre la "Terminal" desde el menú de aplicaciones.
3. Una vez que estés en la terminal o símbolo del sistema, puedes utilizar "pip" para instalar paquetes de Python. Por ejemplo, puedes escribir "pip install nombre-del-paquete" para instalar un paquete de Python. Asegúrate de que Python y "pip" estén correctamente instalados y configurados en tu sistema para que esto funcione.
4. Es importante asegurarte de que la carpeta de instalación de Python y la carpeta "Scripts" (en Windows) estén en tu variable de entorno PATH para que puedas ejecutar "pip".

Regenerate

Después de aclarar esto se genera el primer código para poder crear el juego de la culebrita, el cual se puede complementar agregando diferentes elementos.

A continuación, se proporciona un código de ejemplo para un juego de la culebrita en Python utilizando Pygame. Asegúrate de tener imágenes para la cabeza de la serpiente (`head.png`), el cuerpo de la serpiente (`body.png`), y la comida (`food.png`) en la misma carpeta que tu script.

```
python
import pygame
import sys
import random

# Inicialización de Pygame
```

Copy code

```

pygame.init()

# Configuración de la pantalla
ancho, alto = 600, 400
ventana = pygame.display.set_mode((ancho, alto))
pygame.display.set_caption("Culebrita")

# Colores
blanco = (255, 255, 255)
verde = (0, 255, 0)

# Tamaño de la serpiente y la comida
tamaño = 20

# Inicialización de la posición de la serpiente
x, y = ancho // 2, alto // 2
vel_x, vel_y = 0, 0

# Inicialización de la comida
comida_x = random.randrange(0, ancho - tamaño, tamaño)
comida_y = random.randrange(0, alto - tamaño, tamaño)

# Longitud de la serpiente y su cuerpo
longitud = 1
cuerpo = []

# Velocidad de la serpiente
velocidad = 15

# Función para mostrar la puntuación
def mostrar_puntuacion(puntuacion):
    fuente = pygame.font.Font(None, 36)
    texto = fuente.render(f"Puntuación: {puntuacion}", True, blanco)
    ventana.blit(texto, (10, 10))

# Bucle principal del juego
juego_activo = True
puntuacion = 0

while juego_activo:
    for evento in pygame.event.get():
        if evento.type == pygame.QUIT:
            juego_activo = False

        if evento.type == pygame.KEYDOWN:
            if evento.key == pygame.K_LEFT and vel_x == 0:
                vel_x = -tamaño
                vel_y = 0
            if evento.key == pygame.K_RIGHT and vel_x == 0:
                vel_x = tamaño
                vel_y = 0
            if evento.key == pygame.K_UP and vel_y == 0:
                vel_x = 0
                vel_y = -tamaño

```

```

        if evento.key == pygame.K_DOWN and vel_y == 0:
            vel_x = 0
            vel_y = tamaño

    # Mover la serpiente
    x += vel_x
    y += vel_y

    # Comprobar si la serpiente come la comida
    if x == comida_x and y == comida_y:
        puntuacion += 1
        comida_x = random.randrange(0, ancho - tamaño, tamaño)
        comida_y = random.randrange(0, alto - tamaño, tamaño)
        longitud += 1

    # Dibujar la serpiente y su cuerpo

    cabeza = []
    cabeza.append(x)
    cabeza.append(y)
    cuerpo.append(cabeza)
    if len(cuerpo) > longitud:
        del cuerpo[0]

    # Comprobar si la serpiente se choca consigo misma
    for segmento in cuerpo[:-1]:
        if segmento == cabeza:
            juego_activo = False

    # Dibujar la comida
    ventana.fill(blanco)
    for segmento in cuerpo:
        pygame.draw.rect(ventana, verde, [segmento[0], segmento[1], tamaño,
mostrar_puntuacion(puntuacion)

pygame.draw.rect(ventana, verde, [comida_x, comida_y, tamaño, tamaño])

pygame.display.update()

    # Limitar la velocidad del juego
    pygame.time.Clock().tick(velocidad)

# Cerrar Pygame y salir del juego
pygame.quit()
sys.exit()

```

Este código crea un juego de la culebrita utilizando la biblioteca Pygame. La serpiente se mueve por la pantalla y crece cuando come comida. El objetivo es obtener la mayor puntuación posible sin chocar con los bordes de la pantalla o consigo misma. La puntuación se muestra en la esquina superior izquierda de la pantalla.

Asegúrate de tener instalada la biblioteca Pygame y de tener las imágenes necesarias para el juego. Puedes personalizar el juego, agregar música, sonidos o cualquier otra funcionalidad adicional para hacerlo más avanzado y divertido.

Al tener el código base, se empezó a complementar con ayuda de ChatGPT, lo primero fue agregar las imágenes que dan forma a la cabeza y cuerpo de la culebra, además de la comida. Para ellos fue necesario conseguir imágenes en formato PNG. Además, fue necesario aclarar como debían estar dispuestas estos elementos.



¿Las imágenes para la culebrita deben estar dentro de una carpeta especial?



Para que el juego de la culebrita funcione correctamente, debes asegurarte de que las imágenes (head.png, body.png y food.png) estén en la misma carpeta que tu script de Python (el archivo .py). Aquí hay un resumen de cómo organizar tus archivos:



1. **Estructura de archivos:**

- Tu script de Python (por ejemplo, "culebrita.py") debe estar en una carpeta.
- Las imágenes (head.png, body.png y food.png) deben estar en la misma carpeta que tu script de Python.

La estructura de archivos debería verse así:

css

Copy code

```
mi_carpeta/  
├─ culebrita.py  
├─ head.png  
├─ body.png  
└─ food.png
```

Es importante que las imágenes estén en la misma ubicación que tu script Python o que especifiques rutas relativas en el código para que Pygame pueda cargar las imágenes correctamente.

Dentro de tu script Python, puedes cargar las imágenes de la siguiente manera (asumiendo que las imágenes están en la misma carpeta que tu script):

Regenera

python

Copy code

```
head_image = pygame.image.load("head.png")  
body_image = pygame.image.load("body.png")  
food_image = pygame.image.load("food.png")
```

Con esta estructura de archivos y el código adecuado, el juego de la culebrita debería poder cargar las imágenes correctamente y funcionar como se espera. Si las imágenes se encuentran en una ubicación diferente, debes proporcionar la ruta completa o relativa correcta en el código para que Pygame pueda acceder a ellas.



¿Que otras recomendaciones me puedes hacer para que las imágenes funcionen correctamente? Esto ya que no están siendo utilizadas por el código en python.



Para asegurarte de que las imágenes funcionen correctamente en tu código Python, aquí hay algunas recomendaciones:



1. Asegúrate de tener las imágenes de la cabeza, el cuerpo y la comida en el mismo directorio que tu script, como se mencionó anteriormente.
2. Utiliza la función `pygame.image.load()` para cargar las imágenes. Por ejemplo:

python

Copy code

```
head_image = pygame.image.load("head.png")
body_image = pygame.image.load("body.png")
food_image = pygame.image.load("food.png")
```

1. En lugar de dibujar rectángulos, utiliza la función `blit` para mostrar las imágenes en la pantalla. Por ejemplo:

python

Copy code

```
# Para la cabeza de la serpiente
ventana.blit(head_image, (x, y))

# Para el cuerpo de la serpiente
for segmento in cuerpo:
    ventana.blit(body_image, (segmento[0], segmento[1]))

# Para la comida
ventana.blit(food_image, (comida_x, comida_y))
```

1. Asegúrate de que las imágenes tengan las dimensiones correctas y se ajusten a la cuadrícula del juego.

Haciendo estos cambios, el juego debería mostrar las imágenes de la cabeza, el cuerpo y la comida en lugar de los rectángulos. Asegúrate de que las imágenes sean archivos PNG válidos y estén en la ubicación correcta. Además, considera ajustar las dimensiones tamaño de las imágenes para que se adapten al juego de la culebrita.

Regenerate

Con el fin de tener incluidas las imágenes se le pidió nuevamente a la IA que generara el código pero esta vez teniendo en cuentas las recomendaciones recibidas anteriormente.



Por favor corrige el código anterior para que pueda cumplir con estas recomendaciones y de esta manera no se presenten errores con las imágenes.



Aquí tienes el código corregido para el juego de la culebrita en Pygame que utiliza imágenes en lugar de rectángulos para representar la serpiente y la comida:

Asegúrate de que las imágenes "head.png", "body.png" y "food.png" estén en la misma carpeta que el script Python.

python Copy code

```
import pygame
import sys
import random

# Inicialización de Pygame
pygame.init()

# Configuración de la pantalla
ancho, alto = 600, 400

ventana = pygame.display.set_mode((ancho, alto))
pygame.display.set_caption("Culebrita")

# Colores
blanco = (255, 255, 255)

# Tamaño de la serpiente y la comida
tamaño = 20

# Inicialización de la posición de la serpiente
x, y = ancho // 2, alto // 2
vel_x, vel_y = 0, 0

comida_x = random.randrange(0, ancho - tamaño, tamaño)
comida_y = random.randrange(0, alto - tamaño, tamaño)

# Cargar imágenes
head_image = pygame.image.load("head.png")
body_image = pygame.image.load("body.png")
food_image = pygame.image.load("food.png")

# Longitud de la serpiente y su cuerpo
longitud = 1
cuerpo = []

# Velocidad de la serpiente
velocidad = 15

# Función para mostrar la puntuación
def mostrar_puntuacion(puntuacion):
    fuente = pygame.font.Font(None, 36)
    texto = fuente.render(f"Puntuación: {puntuacion}", True, blanco)
    ventana.blit(texto, (10, 10))
```

Regenerate


```

# Bucle principal del juego
juego_activo = True
puntuacion = 0

while juego_activo:
    for evento in pygame.event.get():
        if evento.type == pygame.QUIT:
            juego_activo = False

        if evento.type == pygame.KEYDOWN:
            if evento.key == pygame.K_LEFT and vel_x == 0:
                vel_x = -tamaño
                vel_y = 0
            if evento.key == pygame.K_RIGHT and vel_x == 0:
                vel_x = tamaño
                vel_y = 0
            if evento.key == pygame.K_UP and vel_y == 0:
                vel_x = 0
                vel_y = -tamaño
            if evento.key == pygame.K_DOWN and vel_y == 0:
                vel_x = 0
                vel_y = tamaño

    # Mover la serpiente
    x += vel_x
    y += vel_y

    # Comprobar si la serpiente come la comida
    if x == comida_x and y == comida_y:
        puntuacion += 1
        comida_x = random.randrange(0, ancho - tamaño, tamaño)
        comida_y = random.randrange(0, alto - tamaño, tamaño)
        longitud += 1

    # Dibujar la serpiente y su cuerpo
    cabeza = [x, y]

    cuerpo.append(cabeza)
    if len(cuerpo) > longitud:
        del cuerpo[0]

    # Comprobar si la serpiente se choca consigo misma
    for segmento in cuerpo[:-1]:
        if segmento == cabeza:
            juego_activo = False

    # Dibujar la comida y la serpiente
    ventana.fill(blanco)
    for segmento in cuerpo:
        ventana.blit(body_image, (segmento[0], segmento[1]))
    ventana.blit(head_image, (x, y))
    ventana.blit(food_image, (comida_x, comida_y))
    mostrar_puntuacion(puntuacion)

    pygame.display.update()

```

```
# Limitar la velocidad del juego
pygame.time.Clock().tick(velocidad)

# Cerrar Pygame y salir del juego
pygame.quit()
sys.exit()
```

Este código utiliza las imágenes "head.png", "body.png" y "food.png" para representar visualmente la culebrita y la comida en el juego. Asegúrate de que las imágenes tengan las dimensiones adecuadas y estén en la misma carpeta que el script para que el juego funcione correctamente.

Después de correr el código completo se encontró un problema, ya que las imágenes eran demasiado grandes tanto que tapaban completamente la pantalla, por esto se le pidió una alternativa para escalar las imágenes dentro del mismo código.



El tamaño de las imágenes es demasiado grande y tapa la pantalla completa, ¿Cómo puedo escalar las imágenes para que tomen el tamaño adecuado para las dimensiones del juego?



Para reducir el tamaño de las imágenes en Pygame, puedes escalar las imágenes cargadas utilizando la función `pygame.transform.scale()`. Esta función te permite cambiar las dimensiones de una imagen de acuerdo a las nuevas dimensiones deseadas.

Aquí tienes un ejemplo de cómo puedes modificar el código para escalar las imágenes de la cabeza, el cuerpo y la comida:

```
python Copy code

# Cargar imágenes
head_image = pygame.image.load("head.png")
body_image = pygame.image.load("body.png")
food_image = pygame.image.load("food.png")

# Escalar las imágenes a un nuevo tamaño (por ejemplo, la mitad del tamaño original)
nuevo_tamaño = (10, 10) # Cambiar a las dimensiones deseadas
head_image = pygame.transform.scale(head_image, nuevo_tamaño)
body_image = pygame.transform.scale(body_image, nuevo_tamaño)
food_image = pygame.transform.scale(food_image, nuevo_tamaño)
```

Regenerar

En este ejemplo, hemos definido `nuevo_tamaño` como (10, 10), lo que significa que estamos reduciendo el tamaño de las imágenes a 10x10 píxeles. Puedes ajustar `nuevo_tamaño` a las dimensiones que desees para las imágenes.

Asegúrate de que el nuevo tamaño sea apropiado para tu juego y que las imágenes sigan siendo reconocibles y adecuadas para su uso en el juego de la culebrita.

Modificando las imágenes de esta manera, reducirás su tamaño y las adaptarás para su uso en el juego. Ten en cuenta que cambiar drásticamente el tamaño de las imágenes puede afectar su calidad visual, así que es importante encontrar un equilibrio adecuado entre el tamaño y la apariencia de las imágenes en el juego.

El código principal se debía seguir modificando ya que los colores originales eran los mismos tanto para el fondo como para la puntuación por lo cual no se podía ver los resultados del usuario, para ellos se le pidió ChatGPT que diera alternativas para cambiar el color de la puntuación a negro.



Existe otro problema en cuanto a los colores utilizados en el código, es de suma importancia para el usuario saber la puntuación que lleva pero las letras están del mismo color del fondo del juego y por eso no se puede observar. ¿Cómo puedo cambiar el color de las letras de la puntuación?



El problema que mencionas con el color del puntaje se debe a que el color de la fuente utilizada para mostrar la puntuación en el juego es el mismo que el fondo, lo que hace que el texto sea prácticamente invisible.

Para solucionar este problema, puedes cambiar el color del texto que se muestra en la puntuación. Puedes definir un color diferente para el texto, como el color negro, para asegurarte de que sea visible contra el fondo blanco. Aquí te muestro cómo hacerlo:

```
python Copy code  
  
# Función para mostrar la puntuación  
def mostrar_puntuacion(puntuacion):  
    # Cambia el color del texto a negro (0, 0, 0)  
    fuente = pygame.font.Font(None, 36)  
    texto = fuente.render(f"Puntuación: {puntuacion}", True, (0, 0, 0))  
    ventana.blit(texto, (10, 10))
```

Regenerate

En este caso, hemos cambiado el color del texto a (0, 0, 0), que representa el color negro en RGB. Puedes ajustar el color del texto a cualquier otro valor de color RGB que desees para que sea visible en el fondo blanco.

Con este cambio, el texto de la puntuación debería ser visible y legible en el juego.

También se sugirió cambiar la ubicación del letrero de puntuación.



La ubicación de la puntuación es muy incómoda ya que tapa parte de la pantalla, además que es un lugar poco estratégico para que el jugador pueda estar viendo su puntaje. ¿Cómo podemos cambiar la posición del letrero de la puntuación, para que este en el centro superior de la pantalla?



Claro, para ubicar el letrero en la parte superior central de la pantalla, puedes modificar la posición del texto. Aquí está el código ajustado:

```
def mostrar_puntuacion(puntuacion):
    # Utilizar el tipo de letra original y cambiar el color del texto a negro
    fuente = pygame.font.Font(None, 36)
    texto = fuente.render(f"Puntuación: {puntuacion}", True, negro)

    # Obtener las dimensiones del texto renderizado
    texto_rect = texto.get_rect()

    # Centrar el texto en la parte superior de la pantalla
    texto_rect.midtop = (ancho // 2, 10)

    # Dibujar el texto en la ventana
    ventana.blit(texto, texto_rect)
```

En este código, he ajustado la posición del texto para que esté centrado en la parte superior de la pantalla. Puedes ajustar la posición según tus preferencias si es necesario.

Finalmente, se le añadió al código principal la música correspondiente a cada etapa del juego, para ello se tuvieron que llamar diferentes librerías de pygame.



Este tipo de juegos tiene como característica la música que los acompaña durante la partida, ¿Como es posible agregarle música especial al juego dependiendo del momento de este, como lo es el inicio, durante el juego y cuando pierdes?



Para agregar diferentes tipos de música a tu juego, puedes utilizar la biblioteca Pygame, que proporciona funciones para trabajar con archivos de sonido. A continuación, te muestro cómo puedes incorporar música al inicio, durante el juego y cuando pierdas en tu juego de la culebrita:

1. Importar Pygame.mixer:

Asegúrate de importar `pygame.mixer` junto con `pygame` al principio de tu script.

```
python
import pygame
import pygame.mixer
```

Copy code

2. Inicializar Pygame.mixer:

Antes de utilizar la funcionalidad de mezcla de sonido, debes inicializar Pygame.mixer.

```
python
pygame.mixer.init()
```

Copy code

3. Cargar y Reproducir Música al Inicio:

Puedes cargar y reproducir la música al inicio de tu juego. Asegúrate de tener un archivo de música, como "inicio.mp3", en la misma carpeta que tu script.

```
python
pygame.mixer.music.load("inicio.mp3")
pygame.mixer.music.play(-1) # -1 para reproducir en bucle
```

Copy code

4. Reproducir Música Durante el Juego:

Puedes reproducir una pista diferente durante el juego. Esto podría ser una música más animada o relacionada con la acción del juego.

python

Copy code

```
pygame.mixer.music.load("juego.mp3")
pygame.mixer.music.play(-1)
```

5. Reproducir Música al Perder:

Cuando el jugador pierde, puedes cambiar la música a una pista diferente, por ejemplo, una música más intensa o una melodía de "game over".

python

Copy code

```
pygame.mixer.music.load("game_over.mp3")
pygame.mixer.music.play(0) # 0 para reproducir una vez sin bucle
```

6. Detener la Música:

Al salir del juego, es buena práctica detener la música.

python

Copy code

```
pygame.mixer.music.stop()
```

Regenerar

Asegúrate de tener archivos de música ("inicio.mp3", "juego.mp3", "game_over.mp3") en la misma carpeta que tu script, y ajusta los nombres de archivo según sea necesario.

Para finalizar el código se le pidió a ChatGPT reconstruir el código con todas las partes que se le modificaron, además de las partes que se le agregaron al juego como la música y las imágenes.



Por favor puedes crear todo el código completo juntando los códigos para el sonido, las imágenes y demás características que se modificaron con el fin de complementar el juego.



Aquí está el código completo con estas incorporaciones:

python

Copy code

```
import pygame
import pygame.mixer
import sys
import random

# Inicialización de Pygame y Pygame.mixer
pygame.init()
pygame.mixer.init()

# Configuración de la pantalla
ancho, alto = 600, 400
ventana = pygame.display.set_mode((ancho, alto))
pygame.display.set_caption("Culebrita")

# Colores
blanco = (255, 255, 255)
negro = (0, 0, 0)

# Tamaño de la serpiente y la comida
tamaño = 20

# Inicialización de la posición de la serpiente
x, y = ancho // 2, alto // 2
vel_x, vel_y = 0, 0
```

Regenerar

```

# Inicialización de la comida
comida_x = random.randrange(0, ancho - tamaño, tamaño)
comida_y = random.randrange(0, alto - tamaño, tamaño)

# Cargar imágenes
head_image = pygame.image.load("cabeza.png")
body_image = pygame.image.load("cuerpo.png")
food_image = pygame.image.load("comida.png")

nuevo_tamaño = (25, 25) # Cambiar a las dimensiones deseadas
head_image = pygame.transform.scale(head_image, nuevo_tamaño)
body_image = pygame.transform.scale(body_image, nuevo_tamaño)
food_image = pygame.transform.scale(food_image, nuevo_tamaño)

# Longitud de la serpiente y su cuerpo
longitud = 1
cuerpo = []

# Velocidad de la serpiente
velocidad = 15

# Función para mostrar la puntuación
def mostrar_puntuacion(puntuacion):
    # Utilizar el tipo de letra original y cambiar el color del texto a negro
    fuente = pygame.font.Font(None, 36)
    texto = fuente.render(f"Puntuación: {puntuacion}", True, negro)

    # Obtener las dimensiones del texto renderizado
    texto_rect = texto.get_rect()

```

```

    texto_rect = texto.get_rect()

    # Centrar el texto en la parte superior de la pantalla
    texto_rect.midtop = (ancho // 2, 10)

    # Dibujar el texto en la ventana
    ventana.blit(texto, texto_rect)

# Cargar y Reproducir Música al Inicio
pygame.mixer.music.load("inicio.mp3")
pygame.mixer.music.play(-1) # -1 para reproducir en bucle

# Bucle principal del juego
juego_activo = True
puntuacion = 0

while juego_activo:
    for evento in pygame.event.get():
        if evento.type == pygame.QUIT:
            juego_activo = False

        if evento.type == pygame.KEYDOWN:
            if evento.key == pygame.K_LEFT and vel_x == 0:
                vel_x = -tamaño
                vel_y = 0

```

```

        if evento.key == pygame.K_RIGHT and vel_x == 0:
            vel_x = tamaño
            vel_y = 0
        if evento.key == pygame.K_UP and vel_y == 0:
            vel_x = 0
            vel_y = -tamaño
        if evento.key == pygame.K_DOWN and vel_y == 0:
            vel_x = 0
            vel_y = tamaño

# Mover la serpiente
x += vel_x
y += vel_y

# Comprobar si la serpiente come la comida
if x == comida_x and y == comida_y:
    puntuacion += 1
    comida_x = random.randrange(0, ancho - tamaño, tamaño)
    comida_y = random.randrange(0, alto - tamaño, tamaño)
    longitud += 1

# Dibujar la serpiente y su cuerpo
cabeza = [x, y]
cuerpo.append(cabeza)
if len(cuerpo) > longitud:
    del cuerpo[0]

# Comprobar si la serpiente se choca consigo misma
for segmento in cuerpo[:-1]:
    if segmento == cabeza:
        juego_activo = False

# Cambiar el color de fondo a blanco (255, 255, 255)
ventana.fill( blanco )

# Dibujar la comida y la serpiente
for segmento in cuerpo:
    ventana.blit( body_image, (segmento[0], segmento[1]) )
ventana.blit( head_image, (x, y) )
ventana.blit( food_image, (comida_x, comida_y) )
mostrar_puntuacion( puntuacion )

pygame.display.update()

# Limitar la velocidad del juego
pygame.time.Clock().tick( velocidad )

# Reproducir Música al Perder
pygame.mixer.music.load( "game_over.mp3" )
pygame.mixer.music.play( 0 ) # 0 para reproducir una vez sin bucle

# Esperar un momento antes de cerrar la ventana
pygame.time.delay( 3000 )

# Detener la Música y Cerrar Pygame
pygame.mixer.music.stop()
pygame.quit()
sys.exit()

```

Aunque no se le agregó al código final, si pe pregunto si había la posibilidad de introducirle al juego niveles de dificultad mediante el usuario fuera ganando puntos, para lo cual ChatGPT contesto que sí.



Lo ideal seria poder implementar un nivel de dificultad en este juego, ¿Es posible crear niveles los cuales aumentan según la puntuación de usuario?



Sí, es posible agregar niveles a tu juego de la culebrita para aumentar la complejidad y desafío a medida que el jugador avanza. Aquí hay una guía general sobre cómo podrías implementar niveles en tu juego:

1. Define Estructura de Niveles:

Decide cómo quieres estructurar tus niveles. Podrías tener un archivo de configuración que contenga información sobre cada nivel, como la velocidad de la serpiente, la cantidad de comida requerida, el tamaño del área de juego, etc.

2. Crea Funciones para Cargar Niveles:

Implementa funciones para cargar la información del nivel actual y aplicar las configuraciones correspondientes.

3. Aumenta la Dificultad con Cada Nivel:

Incrementa la dificultad a medida que el jugador avanza. Esto podría incluir aumentar la velocidad de la serpiente, reducir el tiempo para completar el nivel, o aumentar la cantidad de comida necesaria para avanzar.

4. Diseña Nuevos Obstáculos o Desafíos:

Considera agregar nuevos elementos al juego en niveles posteriores para mantener el interés del jugador. Esto podría incluir obstáculos en el camino de la serpiente o patrones de movimiento más complejos para la comida.

5. Maneja la Transición entre Niveles:

Determina cómo el juego pasa de un nivel al siguiente. Puedes mostrar un mensaje de éxito al completar un nivel y luego cargar la configuración del siguiente nivel.

Regenerat

6. Implementa un Sistema de Puntuación por Niveles:

Lleva un registro de la puntuación del jugador en cada nivel. Podrías calcular la puntuación total al final del juego sumando las puntuaciones de cada nivel.

A continuación, se presenta un ejemplo simple de cómo podrías estructurar tu código para incluir niveles. Ten en cuenta que esto es solo un esquema básico, y podrías adaptarlo según tus necesidades específicas:

```
python Copy code

import pygame
import sys
import random

# ... (código de inicialización y configuración)

# Función para cargar un nuevo nivel
def cargar_nivel(nivel):
    # Lógica para cargar configuraciones específicas del nivel
    # Puedes ajustar la velocidad, cantidad de comida, obstáculos, etc.

# Bucle principal del juego
juego_activo = True
nivel_actual = 1
puntuacion_total = 0
```



```

while juego_activo:
    # ... (código del juego)

    # Comprobar si se completa el nivel actual
    if puntuacion >= comida_necesaria_para_siguiente_nivel:
        nivel_actual += 1
        puntuacion_total += puntuacion
        cargar_nivel(nivel_actual)

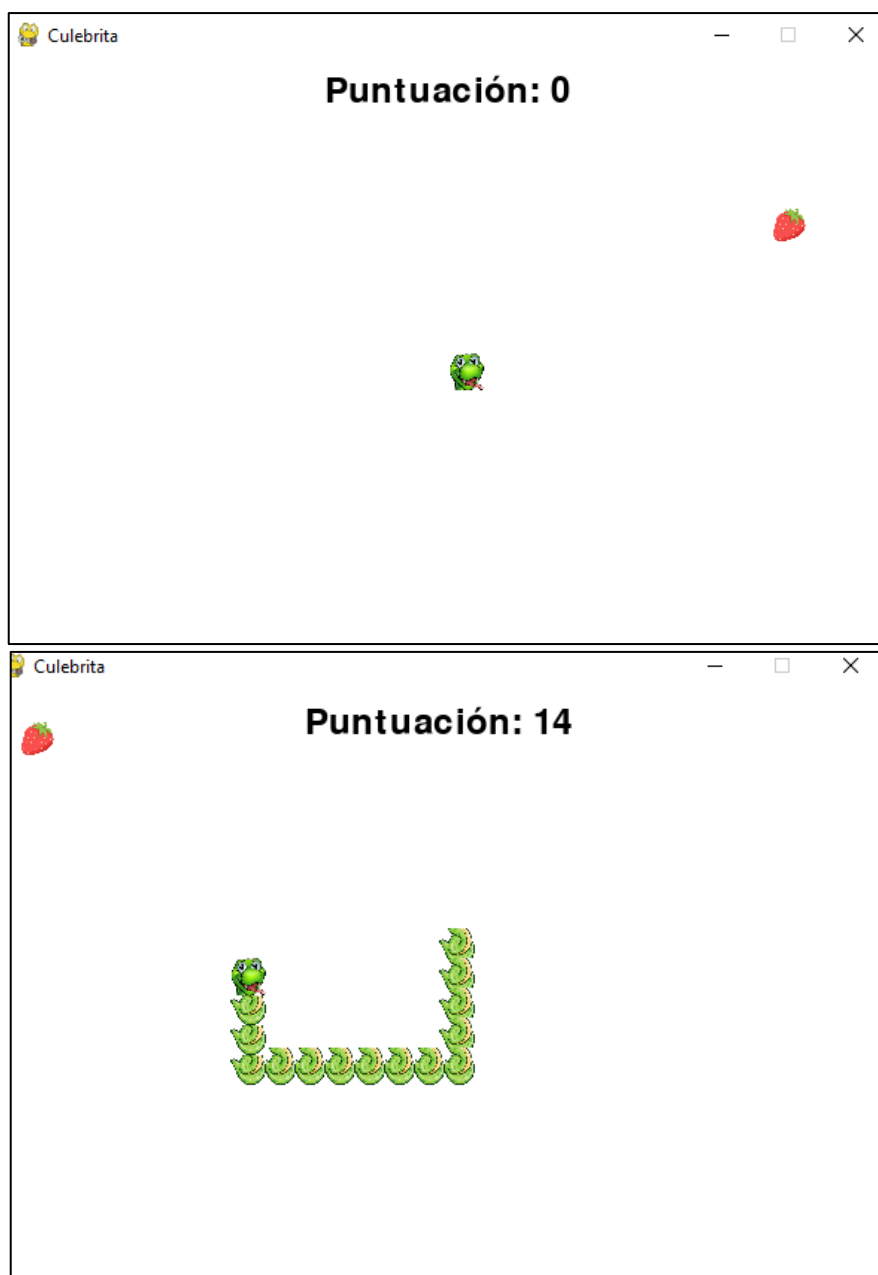
    # ... (resto del código del juego)

# Al salir del bucle principal (por ejemplo, cuando el jugador pierde)
pygame.quit()
sys.exit()

```

Este es solo un ejemplo básico para darte una idea. Puedes expandir y personalizar esta estructura según tus necesidades específicas y la complejidad que deseas para tu juego.

Al correr el código el juego se ve de la siguiente forma:



Cabe recordar que para que este código funcione se deben descargar todos los documentos de este repositorio, también se pueden cambiar según el gusto del programador, además se debe tener instalada la librería de pygame.

V. Lecciones Aprendidas

En la programación ya sea de juegos o de programas más avanzados se deben realizar códigos extremadamente complejos en los procesos de diseño para aplicaciones, por eso al poder implementar la ingeniería de prompts ayuda a minimizar los tiempos de programación, además ayuda a realizar códigos con una alta precisión lo cual es de suma importancia para garantizar el funcionamiento de los programas. Cabe resaltar que para poder llegar a ese punto de precisión es necesario entender el funcionamiento de la IA, ya que de eso depende la certeza de las respuestas que esta proporcione.

VI. Conclusiones

Los resultados que se obtuvieron en este trabajo fueron los esperados, ya que gracias a la ayuda de ChatGPT se logró crear un código que ayudara a un ingeniero a programar un juego básico como la culebrita por medio de Python, claro está que en el proceso se presentaron diferentes dificultades, pero al realizar las correcciones en cuanto a la forma de solicitar información a la IA y la reestructuración del código general se logró implementar la ingeniería de prompts en un tema de alta complejidad de la programación.

Camilo Andrés Bohórquez Piñeros estudiante de Ingeniería Mecánica.