# Proyecto: Desarrollo del juego Snake utilizando Pygame

**Descripción General:**

El objetivo de este proyecto es desarrollar una versión moderna y funcional del clásico juego de **Snake** utilizando la librería **Pygame** en Python. Los estudiantes deberán estructurar su código utilizando principios de **Programación Orientada a Objetos (POO)**, manejar correctamente eventos y animaciones, e implementar características avanzadas como el crecimiento de la serpiente, colisiones, puntuaciones, y niveles de dificultad. Como extensión opcional, se puede añadir una IA que juegue contra el usuario o colabore con él, creando un modo de juego con varias serpientes en pantalla.

**Objetivos del Proyecto:**

1. **Comprender el uso de Pygame**: Los estudiantes aprenderán cómo utilizar esta librería para crear una interfaz gráfica, manejar eventos, detectar colisiones y generar animaciones.
2. **Estructurar el proyecto con POO**: Organizar el código utilizando clases para encapsular el comportamiento de la serpiente, la comida, y otros elementos del juego.
3. **Implementar mecánicas de juego clásicas**: Como el crecimiento de la serpiente al comer, la detección de colisiones con el propio cuerpo o las paredes, y el fin del juego al ocurrir una colisión.
4. **Sistema de puntuación**: Añadir un marcador en pantalla que incremente conforme la serpiente crezca.
5. **Niveles de dificultad**: Permitir al jugador seleccionar entre distintos niveles de dificultad (por ejemplo, aumentando la velocidad de la serpiente o el tamaño del área de juego).
6. **Extensión opcional**: Integrar una inteligencia artificial que compita o colabore con el jugador (multi-serpiente).

**Instrucciones Detalladas:**

**1. Instalación de Pygame y Configuración Inicial**

* Los estudiantes deben instalar la librería **Pygame** utilizando pip install pygame.
* Iniciar un archivo principal snake\_game.py que servirá como punto de entrada para el juego.
* Crear una ventana de juego con una resolución específica (p. ej., 600x600 píxeles) utilizando pygame.display.set\_mode().
* Establecer el bucle principal del juego donde se controlarán los eventos, como la actualización de la pantalla y la lectura de las teclas del usuario.

**2. Estructura del Código con POO**

* Crear una clase Serpiente que encapsule los atributos y comportamientos de la serpiente:
  + **Atributos**: Posición inicial, tamaño (lista de segmentos), velocidad.
  + **Métodos**:
    - mover(): Actualiza la posición de la serpiente según la dirección en la que se mueve.
    - dibujar(): Dibuja la serpiente en la pantalla.
    - colisionar(): Detecta colisiones con las paredes o con su propio cuerpo.
* Crear una clase Comida para manejar la generación de la comida en posiciones aleatorias en la pantalla:
  + **Atributos**: Posición de la comida.
  + **Métodos**:
    - dibujar(): Dibuja la comida en la pantalla.
    - nueva\_posicion(): Coloca la comida en una nueva posición aleatoria cuando la serpiente la come.

**3. Implementación de Mecánicas Clásicas del Juego**

* **Movimiento de la serpiente**: La serpiente debe moverse de forma continua en la dirección en la que se esté desplazando. Utilizar eventos de teclado (pygame.KEYDOWN) para cambiar la dirección (arriba, abajo, izquierda, derecha).
* **Crecimiento de la serpiente**: Cada vez que la serpiente coma comida, se debe añadir un segmento nuevo a su cuerpo.
* **Detección de colisiones**: Implementar la lógica para detectar si la serpiente choca con las paredes o con su propio cuerpo. En caso de colisión, el juego termina.

**4. Sistema de Puntuación**

* Implementar un contador de puntos que aumente cada vez que la serpiente coma comida.
* Mostrar la puntuación en tiempo real en la pantalla utilizando pygame.font.Font() para renderizar el texto.

**5. Niveles de Dificultad**

* Permitir que el jugador seleccione entre diferentes niveles de dificultad antes de empezar el juego.
  + **Fácil**: La serpiente se mueve lentamente y el área de juego es amplia.
  + **Intermedio**: Velocidad de la serpiente moderada.
  + **Difícil**: La serpiente se mueve rápidamente y las colisiones son más difíciles de evitar.
* La dificultad puede ajustarse modificando la velocidad de actualización del juego (pygame.time.Clock().tick()).

**6. Extensión Opcional: Implementación de IA (Multi-serpiente)**

* **Modo competitivo**: Crear una segunda serpiente controlada por una IA que compita contra el jugador por la comida.
  + Implementar un algoritmo sencillo donde la IA busque la comida más cercana y se desplace hacia ella.
  + Manejar colisiones entre las dos serpientes, así como colisiones individuales.
* **Modo colaborativo**: Ambas serpientes (controlada por el jugador y por la IA) cooperan para lograr un objetivo común, como alcanzar cierta longitud combinada o recoger un número determinado de alimentos.

**7. Gráficos y Mejoras Visuales**

* Personalizar los gráficos de la serpiente y la comida, cambiando colores o añadiendo sprites personalizados.
* Añadir efectos visuales sencillos, como animaciones cuando la serpiente crezca o un efecto de "parpadeo" al colisionar.

**8. Finalización del Proyecto**

* Probar el juego para asegurar que todas las funcionalidades (movimiento, crecimiento, colisiones, puntuación, niveles de dificultad) funcionen correctamente.
* Añadir mensajes finales al juego, como "Game Over" cuando la serpiente colisiona, y permitir al jugador reiniciar o salir del juego.

**Requisitos del Proyecto:**

1. **Pygame** instalado.
2. Organización del código en al menos dos clases: Serpiente y Comida.
3. Implementación de todas las mecánicas principales del juego.
4. Sistema de puntuación y niveles de dificultad.
5. Extensión opcional: IA controlando una segunda serpiente.

**Evaluación:**

* **Funcionalidad básica** (movimiento de la serpiente, colisiones, puntuación) – 40%
* **Uso de POO** (claridad y organización del código) – 20%
* **Sistema de niveles de dificultad y opciones para el jugador** – 20%
* **Calidad de la extensión opcional (IA o multi-serpiente)** – 10%
* **Presentación del proyecto y gráficos** – 10%