

Documentación Técnica del Proyecto

Juego Snake con Decoradores y Adaptadores

Juan David Mayorga López – 20232020116
Samuel Andrés Barrera Pulido – 20232020156
Mariam Betin Escobar – 20232020300

Para la clase de Modelos de Programación
Profesor: Alejandro Daza
October 8, 2025

Contents

1	Resumen del Proyecto	2
2	Arquitectura General	2
3	Patrones de Diseño Utilizados	2
4	Descripción de Componentes	2
4.1	Colores y Paletas	2
4.2	Clase Snake	2
4.3	Clase Food	3
4.4	Clase Scoreboard	3
4.5	Decoradores	3
5	Flujo de Ejecución	3
6	Conclusión	3

1 Resumen del Proyecto

Este proyecto consiste en la implementación del clásico juego *Snake* (serpiente) utilizando la librería `turtle` de Python para gráficos. El juego incorpora un sistema de **decoradores** que modifican dinámicamente aspectos visuales como el color del fondo y de la serpiente, o incluso la reducción del cuerpo de la serpiente. Como también se aplica un **adapter** que permite al jugador selección entre diferentes modos de control: teclado y ratón. Se aplican conceptos de diseño de software para organizar el código en clases, decoradores y adaptadores, asegurando modularidad y facilidad de mantenimiento. El jugador controla una serpiente que debe comer comida para crecer, mientras evita chocar con los bordes o consigo misma. Los colores de la serpiente, la comida y el fondo se eligen aleatoriamente evitando solapamientos para mejorar la experiencia visual.

2 Arquitectura General

El proyecto está organizado en diferentes clases y componentes, que interactúan para gestionar el estado del juego:

- **Clase Game:** Controla el bucle principal del juego, las actualizaciones de pantalla, la interacción entre la serpiente, la comida y el marcador. Gestiona los decoradores que modifican el aspecto visual.
- **Clase Snake:** Representa la serpiente. Controla el movimiento, crecimiento, color del cuerpo y detección de colisiones.
- **Clase Food:** Representa la comida. Genera posiciones y colores aleatorios, asegurando que no se solapen con la serpiente ni el fondo.
- **Clase Scoreboard:** Maneja el puntaje y el marcador en pantalla, incluyendo el mensaje de *Game Over*.
- **Decoradores (Decorador y sus subclases):** Aplican modificaciones visuales al juego, como cambiar el color del fondo, modificar el color de la serpiente o reducir el cuerpo, utilizando el patrón decorador para flexibilizar estas funcionalidades.
- **Clase ControlAdapter:** Permiten cambiar el tipo de control del jugador con teclado y ratón.

3 Patrones de Diseño Utilizados

El proyecto utiliza dos patrones de diseño: principalmente:

- **Clase Game: Patron Decorator:** Permite aplicar cambios visuales en diferentes aspectos del juego sin alterar la estructura central. Además, se utiliza la encapsulación y modularización en clases para mantener el código organizado.
- **Patron Adapter:** Permite conectar distintos tipos de entradas (teclado o ratón) a una misma interfaz de control, haciendo el sistema extensible y adaptable a nuevos métodos de interacción.

Ambos patrones se integran de forma complementaria, contribuyendo a una arquitectura flexible y modular.

4 Descripción de Componentes

4.1 Colores y Paletas

Se define una lista de colores posibles para la serpiente, la comida y el fondo, evitando el uso del color negro para mantener visibilidad. Para el fondo, se excluyen colores como blanco y negro, para evitar falta de contraste.

4.2 Clase Snake

- **Atributos:** cabeza (`head`), lista de segmentos (`segments`), color del cuerpo (`body_color`), dirección actual.
- **Métodos:**
 - `move`: mueve la cabeza y actualiza los segmentos del cuerpo.
 - `grow`: añade un nuevo segmento al cuerpo.
 - `change_body_color`: cambia el color del cuerpo de la serpiente.
 - `reset`: reinicia la serpiente después de una colisión.
 - Control de dirección: métodos para cambiar la dirección evitando giros de 180°.

4.3 Clase Food

Gestiona la posición y color de la comida. Se asegura que el color y posición no coincidan con el fondo ni la serpiente para buena visibilidad.

4.4 Clase Scoreboard

Maneja el puntaje actual y el récord, mostrando información en pantalla y el mensaje de *Game Over* cuando sea necesario con un control que permite reiniciar el juego presionando a barra espaciadora después de dicho mensaje.

4.5 Decoradores

- `DecoradorFondo`: cambia el color de fondo evitando colores ya usados.
- `DecoradorSerpiente`: cambia el color del cuerpo de la serpiente.
- `DecoradorReducirCuerpo`: elimina los segmentos actuales de la serpiente (`reset` visual del cuerpo).

Estos decoradores se aplican aleatoriamente cuando la serpiente come comida, creando efectos visuales dinámicos, y cada decorador muestra su activación en el marcador mediante el `update_decorator`.

4.6 Adaptadores de Control

- `ControlAdapter`: Define como controlar el juego con `conectar_controles(game)`
- `KeyboardAdapter`: adapta los controles del teclado al juego.
- `MouseAdapter`: Permite mover la serpiente siguiendo el puntero del raton.
- `ControlSelector`: Muestra una pantalla inicial donde se elige el modo de juego.

Con esta estructura es posible añadir fácilmente nuevos modos de control sin modificar el código base.

5 Flujo de Ejecución

1. El programa inicia creando una instancia de la clase `Game`.
2. El jugador selecciona el tipo de control (teclado o ratón) mediante el `ControlSelector`.
3. Se inicializan la serpiente, la comida, el marcador y los decoradores.
4. Durante el juego, la serpiente se mueve según el modo de control elegido.
5. Al comer, crece, aumenta el puntaje y puede activarse un decorador aleatorio.
6. Si hay colisión con los bordes o con su cuerpo, se muestra el mensaje de *Game Over*.
7. El jugador puede reiniciar presionando la barra espaciadora.
8. El bucle continúa hasta que el usuario cierre la ventana.

6 Conclusión

Este proyecto muestra cómo organizar un juego clásico utilizando programación orientada a objetos, el patrón decorator y adapter para mejorar la flexibilidad visual sin comprometer la estructura base. La implementación con la librería `turtle` permite un desarrollo sencillo y visualmente atractivo, integrando controles, efectos y lógica de juego en un sistema modular.

