Documentación Proyecto II

Eslyn Andrés Jara Calderón

Profesor. Juan Caamaño Alvarez

Universidad Nacional de Costa Rica

Programación II

EIF204

27/05/2025

El presente proyecto posee la capacidad de crear criaturas y modelar un comportamiento, además de simular un entorno donde estos se desarrollen en la búsqueda por supervivencia por recursos. Estas criaturas tendrán diferentes tipos de comportamiento relacionado a su alimentación. Es por esto que para las diferentes cualidades de cada uno o eventos se usarán patrones de diseño implementado en las clases de cada uno de ellos.

Librerías externas

La liberia *Raylib* se usará en este proyecto para manejar el flujo entre ventas para el menú y la simulación, este usará elementos y funciones para pintar los objetos de tipo recurso, criatura y entorno. La librería también permite la creación de botones, campos de texto, y además de cargar elementos de sonido e imágenes.

Recursos

La clase Recurso tendrán su uso mediante otros patrones y serán agrupados, estos aparecerán y desaparecerán en el mapa, estas subclases poseen únicamente la función de dibujarse y ser estáticas para su consumo. La funcionalidad de recurso será otorgar energía y darle vida a las criaturas con su alimentación respectiva.

Criaturas

La clase Criatura es una clase abstracta la cual está contenida en un arreglo dando herencia a múltiples hijos, esta clase maneja la virtualidad de métodos y miembros como la velocidad, posición y textura. Además esta clase aplica los patrones de estrategia para que el vector individualmente los aplique a cada criatura del contenedor.

Patrón *Factory Method (Abstract Factory)*

La clase Criatura será una clase abstracta que servirá como base para tres clases derivadas: Omnívoro, Herbívoro y Carnívoro. Por otro lado, la clase Recurso también será una clase base, de la cual heredarán las siguientes clases: Recurso Agua, RecursoCarne y RecursoPlanta. Las clases derivadas de Recurso implementarán métodos como dibujar y actualizar los cuales permitirán representar y actualizar visualmente estos objetos dentro de una ventana externa utilizando la librería *Raylib*.

El patrón de fábrica abstracta implementa el uso de una clase Fábrica donde se usa la abstracción en métodos para la creación de una nueva criatura en la clase de fábrica concreta. El patrón únicamente se encargará de crear objetos criatura y recurso mediante el uso de los métodos desarrollados en la clase fábrica concreta.

Patron *Singleton*

El patrón está envuelto en el entorno, este aplica los cambios de clima tales como nieve, soleado y lluvia, esta clase busca mediante la interfaz obtener los comandos para poder cambiar el entorno, se encarga gráficamente también de cambiar los fondos y música. Esta clase únicamente se encarga de recibir un comando para ejecutar un nuevo entorno.

Patrón *Observer*

El patrón *Observe*r tiene la finalidad de que los recursos reaccionen a cambios en el entorno. En el proyecto tiene la función de qué dependen del comando en el área de simulación, que registre el nuevo entorno (Singleton) y se modifique su frecuencia de aparición en el área. Este creará cada segundo un recurso aleatoriamente que aparecerá en el campo y será agregado al arreglo de recursos. El patrón únicamente influye en recursos ya que no tiene ninguna reacción en los animales por la caza y búsqueda de recursos.

Patrón Iterator

En este proyecto descarto la idea de usar el patrón de diseño para modificar el estado de los conjuntos o colecciones de elementos. Ya que al ser visual, este se verá reflejado en pantalla y no necesitará un orden de conjuntos para ver el estado de cada criatura o incluso los comportamientos realizados, por ende se verá cada movimiento en tiempo real de cada colección en el área de simulación del proyecto

Patrón *Composite*

El sistema de simulación rechaza también el uso del patrón para la modelación de recursos, los recursos no necesitan una jerarquía ya que cada vez por el tipo de entorno se van generando un recurso y a su vez viven o mueren dependiendo del clima del entorno, además de mostrarse visualmente no necesitará un agrupamiento de tipo compuesto. Las criaturas viven únicamente de los

recursos, no están dependiendo del entorno, los recursos si, las criaturas solo buscan comer para vivir o comer, y dependiendo de su alimentación se reproducen.

Patrón *Strategy*

El patrón Estrategia permite definir unas funciones de manera independiente para cada criatura y recurso, para poder ser aplicado en tiempo real. el proyecto aplica las acciones fundamentales para los seres vivos del entorno: moverse, reproducirse, morir y alimentarse

A. **Estrategia de Movimiento**: Esta estrategia gestiona el desplazamiento de las criaturas contenidas en un arreglo. Las criaturas se mueven aumentando sus coordenadas en los ejes X e Y, y rebotan al alcanzar los bordes del campo, simulando una interacción con los límites del entorno.

B. **Estrategia de reproducción**: Controla el proceso de apareamiento de criaturas. Una criatura puede reproducirse si ha consumido al menos dos alimentos en menos de 10 segundos, lo cual indica un metabolismo acelerado. Al cumplirse esta condición, se genera una nueva criatura en el entorno mediante el patrón *Abstract Factory*.

C. **Estrategia de Muerte**: Controla la hambruna de las criaturas. Si una criatura pasa demasiado tiempo sin alimentarse, llega a morir eventualmente y se transforma en un recurso de alimento estático (carne) que puede ser consumido por criaturas omnívoras y carnívoras.

D. **Estrategia de Alimentación**: Determina si la criatura puede consumir un recurso específico, validando si es adecuado para su tipo. Al alimentarse, la criatura recupera vida y energía, lo que le permite prolongar su existencia y poder mantener su existencia en el entorno.

Patrón *Decorator*

Se descarta la idea del patrón ya que no se trabajó con ninguna habilidad de las criaturas en tiempo real.