CICLO 1: ANÁLISIS Y CORRECCIÓN DE REQUISITOS

Miniciclo 1.1: Evaluación del Proyecto Preliminar

En esta fase se revisa el documento PDF original del proyecto, que fue seleccionado mediante votación. Se analizan:

- La idea general del juego.
- Los modos de juego planteados (Jugador vs Jugador y Machine vs Machine).
- La estructura básica de clases necesaria para implementar la lógica del juego.

Se definen los primeros diagramas de clases (UML) y se documentan los requisitos iniciales, identificando qué componentes deben existir (jugador, enemigos, bloques, niveles, máquinas, etc.) y cómo se relacionan. Este miniciclo sienta la base de la arquitectura orientada a objetos.

Miniciclo 1.2: Adaptación a Nuevos Requisitos

Después de la segunda entrega de código, se modifican gran parte de los requisitos y se agregan nuevos. Esto obliga a:

- Analizar las diferencias entre los requisitos originales y los nuevos.
- Detectar qué clases, métodos y estructuras dejan de ser válidas.
- Refactorizar el código existente para adaptarlo a los nuevos cambios.

En este proceso se reorganizan responsabilidades, se reajustan relaciones entre clases y se aplican principios SOLID para mejorar la mantenibilidad. Se actualizan los diagramas UML y se documentan claramente las modificaciones realizadas.

Miniciclo 1.3: Implementación de Modos de Juego Corregidos

Con la arquitectura corregida, se vuelven a implementar los modos:

- **PvsP (Jugador vs Jugador):**
  Gestión de entrada de dos jugadores, control de turnos o acciones simultáneas, sistema de puntuación para cada jugador, detección de victoria/derrota individual.
- **Machine vs Machine:**
  Implementación de la lógica para que ambas máquinas sean controladas por la computadora. Cada máquina sigue reglas de comportamiento automáticamente (selección de objetivos, movimiento, acciones en el mapa).

Se emplean patrones como Strategy y State para manejar el comportamiento de las máquinas y la evolución del juego.

Este miniciclo garantiza que los modos de juego funcionen de acuerdo con los nuevos requisitos del proyecto.

CICLO 2: EXPANSIÓN DE CONTENIDO DEL JUEGO

Miniciclo 2.1: Implementación del Nuevo Enemigo - Narval

Se añade un nuevo enemigo llamado Narval, cuya característica principal es que cuenta con **una habilidad especial única** respecto a los demás enemigos.

- El Narval hereda de la clase base de enemigos (por ejemplo, Enemy).
- Mantiene propiedades estándar como velocidad, movimiento y puntos al ser derrotado.
- Se implementa una **habilidad especial** exclusiva, que puede activarse bajo ciertas condiciones del juego (por ejemplo, cercanía al jugador, tiempo transcurrido o estado particular del nivel).
- Se programan métodos para la activación y ejecución de esa habilidad especial.

Con esto se refuerza el uso de herencia y polimorfismo, permitiendo integrar al Narval sin romper la lógica ya implementada para otros enemigos.

Miniciclo 2.2: Implementación de Nuevos Bloques

En este miniciclo se agregan únicamente dos tipos concretos de bloques, con efectos muy claros en la jugabilidad:

- **Bloques Indestructibles:**
  Son obstáculos permanentes del escenario que el jugador no puede destruir.
  Sirven para:
  - Definir la estructura fija del nivel.
  - Limitar caminos.
  - Condicionar rutas y estrategias del jugador y de los enemigos.

En términos de código, se implementan como una clase específica (por ejemplo, BlockIndestructible) que hereda de una clase base Block, pero sin lógica de destrucción.

- **Baldosas Calientes:**
  Son bloques especiales que afectan **negativamente** al jugador al pisarlos o

tocarlos. Su consecuencia es la Muerte del jugador
Se puede utilizar un patrón Strategy para que cada baldosa pueda aplicar un efecto distinto sin cambiar la estructura base.

Este miniciclo aumenta la profundidad jugable sin recargar el diseño de nuevos tipos de bloques complejos, centrándose en estos dos bien definidos.

Miniciclo 2.3: Implementación de Tipos de Máquinas

Para el modo Machine vs Machine, se definen diferentes **tipos de máquinas**, cada una con un objetivo particular dentro de la partida.

Cada tipo de máquina:

- Tiene su propia lógica de decisión.
- Evalúa el estado del juego (posición de enemigos, frutas, jugador, etc.).
- Selecciona acciones en función de su objetivo principal.

Se implementa un sistema basado en patrones como Strategy para que cada máquina tenga una estrategia intercambiable sin modificar el resto de la estructura del juego.

CICLO 3: INTEGRACIÓN Y CONFIGURACIÓN DE NIVELES

Miniciclo 3.1: Conexión Entre Niveles

Se implementa el sistema que conecta los niveles entre sí. Para lograrlo:

- Se crea un gestor de niveles encargado de:
    - Cargar el nivel correspondiente.
    - Controlar la transición al siguiente nivel.
    - Registrar si el nivel fue superado o no.
- Se definen las condiciones de:
    - **Victoria:** por ejemplo, recolectar todas las frutas, derrotar a cierto número de enemigos, cumplir un objetivo concreto.
    - **Derrota:** ser atrapado por enemigos, agotar el tiempo, u otras condiciones definidas.

Este miniciclo asegura que el juego no sea solo un nivel aislado, sino una progresión coherente entre niveles.

Miniciclo 3.2: Pantalla Final de Cada Nivel

Al completar (ganar o perder) un nivel, se muestra una pantalla final que:

- Presenta estadísticas del nivel:
    - Puntuación alcanzada.
    - Cantidad de frutas recolectadas.
    - Tiempo empleado u otros parámetros relevantes.
- Ofrece al jugador dos opciones principales:
    - Reiniciar el nivel.
    - Avanzar al siguiente nivel (si fue superado).

Esta pantalla puede incluir además mensajes de feedback, medallas o indicadores de desempeño.
Su implementación puede aprovechar el patrón Observer para que se active automáticamente al cumplirse las condiciones de fin de nivel.

Miniciclo 3.3: Sistema de Configuración de Niveles

Antes de empezar un nivel, se brinda al jugador la posibilidad de configurar ciertos parámetros, por ejemplo:

- Cantidad de enemigos contra los que quiere jugar.
- Tipos de enemigos que aparecerán.
- Cantidad de frutas de cada tipo presentes en el mapa.

El sistema debe:

- Validar que los valores elegidos sean viables (no vaciar completamente el nivel, no saturarlo, etc.).
- Aplicar esas configuraciones al momento de generar el nivel.

Se puede utilizar un patrón Builder para construir niveles personalizados a partir de estas opciones, permitiendo que la configuración del jugador se traduzca de forma limpia en la estructura interna del nivel.

CICLO 4: PULIDO Y MEJORA VISUAL

Miniciclo 4.1: Corrección y Mejora de Animaciones Existentes

En este ciclo se aclara que las **mejoras visuales se limitan únicamente a correcciones** de lo que ya existe, sin añadir nuevas animaciones.

Las tareas principales son:

- Revisar todas las animaciones implementadas:
    - Movimiento del jugador.
    - Recolección de frutas.
    - Transiciones entre estados del personaje y pantallas.
- Detectar problemas:
    - Animaciones bruscas.
    - Fotogramas mal alineados.
    - Saltos visuales.
    - Falta de fluidez.
- Ajustar:
    - Duración de los frames.
    - Orden de los sprites.
    - Transiciones entre una animación y otra.

El objetivo es que las animaciones sean fluidas, coherentes y agradables, manteniendo un rendimiento estable, corrigiendo y optimizando los ya existentes.

Miniciclo 4.2: Verificación de Consistencia Visual

Una vez corregidas las animaciones, se realizan pruebas integrales del juego para asegurar que:

- Todas las animaciones se ven coherentes entre sí.
- No se generan errores visuales al cambiar de nivel o modo de juego.
- Los efectos visuales básicos (movimiento, colisiones visibles, cambios de estado) se perciben claramente por el jugador.

Se ajustan detalles finales si se detectan inconsistencias.

Miniciclo 4.3: Versión Final Pulida para Presentación

En este último miniciclo se prepara el juego para ser mostrado o evaluado:

- Se realiza una batería de pruebas completas.
- Se revisan menús, pantallas finales, configuraciones y niveles.
- Se asegura que:
    - Los bloques indestructibles y baldosas negativas funcionen correctamente.
    - El Narval utilice su habilidad especial como está diseñado.
    - Los modos PvsP y Machine vs Machine operen sin errores.
    - La progresión de niveles y la configuración elegida por el jugador estén correctamente integradas.

El resultado de este ciclo es una versión pulida y estable del juego, lista para presentación o entrega como proyecto final.