# RETROSPECTIVA

POOB

13 DE DICIEMBRE DE 2025

HEIDY ALEJANDRA ORJUELA RAMÍREZ
ANDRÉS FELIPE PINEDA GAITÁN

**¿Cuáles fueron los mini-ciclos definidos? Justificación**

Durante el desarrollo del proyecto se definieron varios mini-ciclos de trabajo, cada uno enfocado en una funcionalidad específica del sistema. Estos mini-ciclos permitieron avanzar de manera incremental y reducir riesgos técnicos.

Los mini-ciclos definidos fueron:

1. **Diseño inicial del dominio**
    - ✓ Definición de entidades principales: Juego, Mapa, Jugador, Helado, Entidad.
    - ✓ Justificación: establecer una base sólida de la lógica del juego antes de la interfaz.
2. **Implementación de mecánicas básicas**
    - ✓ Movimiento de helados, colisiones, recolección de frutas y enemigos.
    - ✓ Justificación: asegurar que el juego fuera funcional desde etapas tempranas.
3. **Desarrollo de la capa de presentación**
    - ✓ Ventanas, panel de juego, HUD y renderizado gráfico.
    - ✓ Justificación: visualizar el estado del juego y validar la interacción usuario-sistema.
4. **Integración de IA y modalidades de juego**
    - ✓ Tipos de IA, selección de perfiles y modos de juego.
    - ✓ Justificación: cumplir los requisitos funcionales y de extensión.
5. **Refactorización y mejoras de diseño**
    - ✓ Separación de responsabilidades, fábricas, jerarquías claras.
    - ✓ Justificación: mejorar mantenibilidad, extensibilidad y calidad del diseño.

Estos mini-ciclos permitieron aplicar el principio XP de entregas frecuentes y funcionales.

**2. Estado actual del proyecto en términos de mini-ciclos**

El proyecto se encuentra en un estado avanzado y estable, con los mini-ciclos principales completados.
Actualmente, el sistema:

- Cumple los requisitos funcionales principales.
- Está preparado para extensiones futuras (nuevos niveles, entidades, IA).
- Presenta una arquitectura clara entre dominio, presentación y control.

Esto se debe a que se priorizó primero la funcionalidad básica y luego la calidad del diseño mediante refactorización continua.

**3. Tiempo total invertido (Horas/Hombre)**

El tiempo estimado invertido en el proyecto fue:

- **Análisis y diseño:** ~10 horas
- **Implementación del dominio:** ~15 horas
- **Capa de presentación y renderizado:** ~15 horas
- **Refactorización y documentación:** ~20 horas

**Total aproximado:**

1. 30 horas Heidy Alejandra Orjuela Ramírez
2. 30 horas Andrés Felipe Pineda Gaitán

**4. Mayor logro del proyecto**

El mayor logro fue la correcta separación entre la lógica del dominio y la capa de presentación, manteniendo una arquitectura limpia y extensible.

Esto permitió:

- Cumplir los principios de diseño.
- Facilitar la incorporación de nuevas entidades y modos de juego.
- Mejorar la comprensión y mantenibilidad del código.

Este logro fue importante porque garantiza que el proyecto pueda evolucionar sin necesidad de reescribir grandes partes del sistema.

**5. Mayor problema técnico y solución**

El mayor problema técnico fue el alto acoplamiento inicial entre el controlador y la lógica del juego, especialmente en la gestión de eventos, IA y colisiones.

Para resolverlo se realizaron las siguientes acciones:

- Refactorización del ControladorJuego, delegando responsabilidades al dominio.
- Encapsulación de comportamientos dentro de las entidades correspondientes.
- Uso de fábricas y jerarquías para reducir dependencias directas.

Esto permitió un flujo de ejecución más claro y menos propenso a errores.

**6. Trabajo en equipo: aciertos y compromisos de mejora**

**Lo que se hizo bien:**

- Buena comunicación y división de tareas.
- Aplicación constante de refactorización.
- Pruebas frecuentes del sistema en ejecución.
- Uso de retroalimentación continua para mejorar el diseño.

**Compromisos de mejora:**

- Documentar mejor desde etapas tempranas.
- Definir pruebas automatizadas desde el inicio.
- Planificar mejor los tiempos para evitar acumulación de tareas al final.

**7. Práctica XP más útil**

La práctica de refactorización continua fue la más útil durante el proyecto.

Esto se debe a que permitió:

- Mejorar el diseño sin afectar la funcionalidad.
- Detectar problemas de diseño tempranamente.
- Adaptar el sistema a nuevos requisitos sin grandes reestructuraciones.

**8. Referencias utilizadas**


- Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994).
  *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.
- Beck, K. (2004).
  *Extreme Programming Explained: Embrace Change*. Addison-Wesley.
- Oracle. (2024).
  *Java Platform, Standard Edition Documentation*.
  https://docs.oracle.com/javase/8/docs/