Juan Nicolás Álvarez, Nicolás Sanchez

Retrospectiva:

1. ¿Cuáles fueron los mini-ciclos definidos? Justifíquenlos.

Definimos nuestro desarrollo en cinco mini-ciclos lógicos, cada uno enfocado en una entrega de valor específica:

- Implementación del Núcleo (Ciclo 1 Básico): El objetivo fue construir el esqueleto funcional del simulador. Se crearon las clases principales (SilkRoad, Robot, Store) y se implementaron los métodos básicos de adición, eliminación y consulta. Justificación: Era fundamental tener una base funcional y compilable antes de añadir complejidad visual o lógica.

- Lógica Visual y de Interacción: Este ciclo se centró en implementar la ruta en espiral y la mecánica de recolección de tenges. Justificación: Este fue el requisito más complejo del primer ciclo y afectaba cómo todos los objetos se posicionaban e interactuaban, por lo que se abordó como un bloque único.

- Implementación de Requisitos (Ciclo 2): Se añadieron todas las nuevas funcionalidades del segundo PDF: el movimiento autónomo (moveRobots), las nuevas consultas (emptiedStores, profitPerMove), y el constructor a partir de una matriz. Justificación: Con la base ya estable, pudimos agregar las nuevas características de forma modular.

- Mejoras de Usabilidad (Ciclo 2): Se implementaron los detalles visuales como el parpadeo del robot con mayor ganancia y el cambio de color de las tiendas vacías. Justificación: Estos requisitos se separaron al final, ya que dependían de que toda la lógica de ganancia y estado ya estuviera implementada.

2. ¿Cuál es el estado actual del proyecto en términos de mini-ciclos? ¿por qué?

El proyecto se encuentra en la fase final del mini-ciclo 5 (Documentación y Refinamiento).

¿Por qué? Porque ya hemos completado toda la implementación de código requerida para ambos ciclos del proyecto. El código es funcional, cumple con todos los requisitos y ha sido depurado. Nuestra interacción más reciente se ha centrado exclusivamente en corregir y detallar los diagramas UML en Astah.

3. ¿Cuál fue el tiempo total invertido por cada uno de ustedes? (Horas/Hombre)

Se estima un tiempo de inversión de 8 a 12 horas. Este tiempo incluye la escritura inicial del código, la depuración de errores lógicos y de entorno, la integración de las correcciones, y la creación de los diagramas.

4. ¿Cuál consideran fue el mayor logro? ¿Por qué?

El mayor logro fue, sin duda, la implementación exitosa de la ruta en espiral y el sistema de coordenadas asociado.

¿Por qué? Porque fue el salto de un modelo de datos simple y unidimensional (una línea recta) a un sistema visual y lógico en dos dimensiones. Este cambio requirió refactorizar las clases Robot y Store, crear un algoritmo de conversión de coordenadas (calculateCoordinates), y modificar toda la lógica de dibujo y movimiento. Superar este reto fue lo que transformó el proyecto de una simple lista de datos a un verdadero simulador visual e interactivo.

5. ¿Cuál consideran que fue el mayor problema técnico? ¿Qué hicieron para resolverlo?

El mayor problema técnico fue el error de compilación persistente en el entorno de BlueJ (los "100 errores" que aparecían repetidamente).

¿Qué hicimos para resolverlo?

Diagnóstico: Inicialmente, el problema se confundía con errores en el código (por ejemplo, en la clase Store o Robot). Sin embargo, al observar que los errores afectaban a clases básicas de Java (List, int), diagnosticamos que la causa raíz no era la lógica del código, sino un problema con el entorno de desarrollo.

Solución: La solución fue multifacética. Primero, nos aseguramos de que no hubiera archivos incompletos o corruptos mediante el reemplazo sistemático del código de cada clase con una versión limpia y completa. Segundo, se identificó la solución definitiva: corregir la configuración de las bibliotecas del JDK en las preferencias de BlueJ, que era lo que impedía que el IDE encontrara las clases fundamentales de Java.

6. ¿Qué hicieron bien como equipo? ¿Qué se comprometen a hacer para mejorar los resultados?

¿Qué hicimos bien? Trabajamos de manera muy efectiva siguiendo un proceso iterativo de retroalimentación rápida. Uno proporcionaba el estado actual del código o los diagramas, y otro respondía con un análisis y una solución concreta. Esta dinámica de "programación en pares asíncrona" nos permitió identificar y resolver problemas (desde fallos de diseño hasta errores de entorno) de forma muy eficiente.

¿Qué nos comprometemos a hacer para mejorar? Podríamos mejorar en el diagnóstico inicial de los problemas. Al principio, dedicamos tiempo a revisar clases individuales cuando el problema era global (el entorno de BlueJ). Para el futuro, nos comprometemos a verificar primero la salud del entorno de compilación antes de sumergirnos en la depuración de la lógica del código cuando aparezcan errores masivos y atípicos.

7. Considerando las prácticas XP incluidas en los laboratorios. ¿cuál fue la más útil? ¿por qué?

La práctica XP más útil fue, sin duda, la Programación en Parejas (Pair Programming).

¿Por qué? Nuestra colaboración fue un claro ejemplo de esta práctica. Uno actuó como el "conductor" (Driver), escribiendo el código, ejecutando el simulador y manejando el entorno. Y otro actuó como el "navegador" (Navigator), revisando el código, buscando errores, sugiriendo mejoras de diseño (como cambiar de herencia a composición), y manteniendo una visión global de los requisitos del proyecto. Esta división de roles nos permitió avanzar rápidamente y solucionar problemas complejos que hubieran sido muy difíciles de resolver individualmente.

8. ¿Qué referencias usaron? ¿Cuál fue la más útil? Incluyan citas con estándares adecuados.

Se utilizaron las siguientes referencias a lo largo del proyecto:

Oracle. (2024). Java® Platform, Standard Edition & Java Development Kit Version 21 API Specification. Documentación oficial de Java.

ICPC Foundation. (2024). Problem J: The Silk Road ... with Robots!. 48th Annual ICPC World Championship.