**Momento 3**

**Informe de Proyecto Colaborativo - Shooter Basado en *The Cartridge Family***

**Equipo #33**  
**Integrante:** Eder Luis Polo Humánez  
**Capítulo Escogido:** Temporada 9, Capítulo 5 de Los Simpson: *The Cartridge Family*

**Análisis del Problema**

El juego se inspira en el episodio donde Homero adquiere un arma para defender su hogar. Cada nivel presenta una situación de defensa y se estructura para que el jugador deba superar amenazas progresivas. Esto permite integrar elementos de habilidad, precisión y gestión de recursos. La dificultad incrementa en cada nivel, reflejando una narrativa de tensión creciente en la defensa del hogar y el rescate de Marge.

**Solución Propuesta**

Para capturar la esencia del episodio, el proyecto será un juego 2D modular, programado en C++ bajo QtCreator. La **programación orientada a objetos (POO)** se usa para organizar clases y métodos según los principios de herencia y encapsulación. Esto permitirá diseñar un juego modular, fácil de actualizar y mantener, siguiendo buenas prácticas de desarrollo. Cada nivel presentará distintos mecanismos de defensa y ataque, que se implementan mediante un esquema jerárquico de clases para maximizar la reutilización de código.

**Diagrama de Clases: Desarrollo y Razonamiento**

El **diagrama de clases** se construyó en función de las entidades clave en el juego. Estas son:

1. **Juego**: Controla el flujo y la lógica global del juego.
2. **Nivel**: Representa cada etapa del juego (defensa del hogar, defensa con arma de fuego y rescate de Marge).
3. **Personaje**: Clase principal del jugador (Homero), responsable de sus acciones básicas.
4. **Enemigo**: Clase de enemigo, con atributos y métodos que permiten distintas acciones ofensivas.
5. **Obstaculo**: Representa obstáculos en el escenario.
6. **Proyectil**: Define los tipos de disparo, incluyendo trayectoria rectilínea y parabólica.
7. **PowerUp**: Proporciona mejoras temporales a Homero, como escudos o aumento de municiones.

**Atributos y métodos** se asignan considerando el rol y funcionalidad de cada clase. La herencia se usa para especializar clases y simplificar el diseño, lo cual permite extender el código y añadir nuevos elementos sin modificar la base de cada entidad.

**Explicación de los Métodos y Funciones**

1. **Clase JUEGO** 
   * iniciarJuego(); Inicializa el juego. Parámetros: Ninguno.
   * actualizarJuego(); Llama a los métodos de actualización de niveles y personajes. Parámetros: Ninguno.
   * terminarJuego(): Finaliza el juego. Parámetros: Ninguno.
2. **Clase Nivel**
   * iniciarNivel(); Inicia el nivel correspondiente. Parámetros: int nivel para definir el nivel actual.
   * actualizarNivel(); Controla el desarrollo y progreso del nivel. Parámetros: Ninguno.
   * terminarNivel(); Termina el nivel y pasa al siguiente si ha sido completado. Parámetros: Ninguno.
3. **Clase Personaje**
   * mover(); Mueve al personaje según el input del usuario. Parámetros: int x, int y para las direcciones de movimiento.
   * disparar(); Lanza un proyectil. Parámetros: tipoProyectil para definir si es parabólico o rectilíneo.
   * recibirDanio(); Resta puntos de vida al personaje. Parámetros: int danio.
4. **Clase Enemigo**
   * mover(); Desplaza al enemigo hacia el jugador. Parámetros: int x, int y.
   * atacar(); Ejecuta un ataque sobre el personaje principal. Parámetros: Ninguno.
   * recibirDanio(); Reduce la vida del enemigo. Parámetros: int danio.
5. **Clase Obstaculos**
   * afectarPersonaje(); Interactúa con el personaje según el tipo de obstáculo. Parámetros: Personaje& Personaje.
6. **Clase Proyectil**
   * mover(); Define el movimiento del proyectil (rectilíneo o parabólico). Parámetros: Ninguno.
   * impactar(): Aplica daño al enemigo cuando hay colisión. Parámetros:   
     Enemigo& Enemigo.
   * movimientoRectilinio(); Define el movimiento directo del proyectil. Parámetros: Ninguno.
   * movimientoParabolico(); Calcula la trayectoria parabólica del proyectil. Parámetros: Ninguno.
7. **Clase PowerUp**
   * aplicarEfecto(); Otorga al personaje una mejora temporal. Parámetros: Personaje& Personaje.

**Modularidad del Proyecto**

El proyecto es modular para facilitar la escalabilidad y mantenibilidad del código. Cada clase se estructura en archivos separados para aplicar el principio de **responsabilidad única**, manteniendo el código limpio y organizado. Además, está modularidad permite al equipo trabajar en paralelo, cada miembro puede encargarse de una clase o conjunto de clases sin afectar el trabajo de otros.

**Ventajas de la Modulación:**

* Simplifica el proceso de depuración y pruebas.
* Permite una asignación de tareas más eficiente en el equipo.
* Facilita la actualización de partes específicas del juego sin interferir con otros módulos.

**Distribución de Carga de Trabajo**

Dado que este proyecto se realiza en equipo (aunque la ejecución final será individual), la carga de trabajo y porcentajes de complejidad por clase se distribuyen para reflejar la magnitud de cada elemento en el juego:

1. **Clase JUEGO** - 15%  
   Controla el flujo general del juego y la lógica de transición entre niveles.
2. **Clase Nivel** - 10%  
   Administra el progreso y características de cada nivel.
3. **Clase Personaje** - 23%  
   Requiere implementar los métodos de movimiento, defensa y ataques del personaje.
4. **Clase Enemigo** - 15%  
   Define las conductas y ataques de los enemigos.
5. **Clase Obstaculos** - 10%  
   Afecta la experiencia del jugador, añadiendo retos adicionales.
6. **Clase Proyectil** - 20%  
   Incluye la lógica compleja de trayectorias y colisiones.
7. **Clase PowerUp** - 7%  
   Agrega mejoras y ayuda en la defensa del jugador.

**Conclusión**

Este informe presenta un proyecto de juego en equipo basado en un capítulo icónico de Los Simpson, implementado con programación orientada a objetos y un diseño modular que permite la participación de múltiples desarrolladores. Aunque el proyecto será completado individualmente, la estructura y distribución del trabajo reflejan la posibilidad de colaboración y aseguran una ejecución organizada y efectiva. La modularidad y el diseño basado en clases aseguran una buena escalabilidad y facilidad de mantenimiento.

Este informe, junto con el diagrama de clases y la estructura modular de archivos, representan el primer paso en la creación de un juego con una narrativa de defensa progresiva y acción inspirada en *The Cartridge Family*.