Nombre: Sanjines Alave Samira Denisse

Carrera: Ingeniería en Tecnologías de la Información y Seguridad

**EXAMEN FINAL**

1. **Descripción de Patrones Aplicados**

Creacionales:

- Prototype: Usado para replicar objetos dinámicos como, bloques, bombas o potenciadores. Se define un prototipo de cada objeto y, al necesitar una nueva instancia, simplemente se clona, lo que ahorra tiempo y mantiene consistencia entre variaciones similares.

- Builder: Permite construir laberintos paso a paso de forma flexible y reutilizable*. UConstructorMapa - Builder* para construir el mapa del juego, *UDirectorNiveles - Director* que orquesta la construcción de niveles y *FDatosMapa - Estructura* de datos del mapa

Estructurales:

- Composite: Organiza el laberinto como una estructura en árbol, donde usaremos una agrupación o jerarquía, permitiendo tratar objetos individuales y composiciones de manera uniforme. Por ejemplo: *ASeccionMapa - Composite* que puede contener otros elementos, *ATile - Leaf (elemento hoja)* que implementa la interfaz y *IIElementoMapa - Interfaz común* para componentes y hojas

- Facade: Oculta la complejidad de construcción y gestión del laberinto detrás de una interfaz clara. Por ejemplo, *UGameplayFacade - Facade* principal que simplifica la interacción con todos los sistemas del juego
  Sistemas encapsulados:
  - Constructor de mapas
  - Director de niveles
  - Gestión de eventos
  - Sistema de VFX
  - Sistema de UI
  - Gestión de bombas, power-ups y enemigos

Comportamiento:

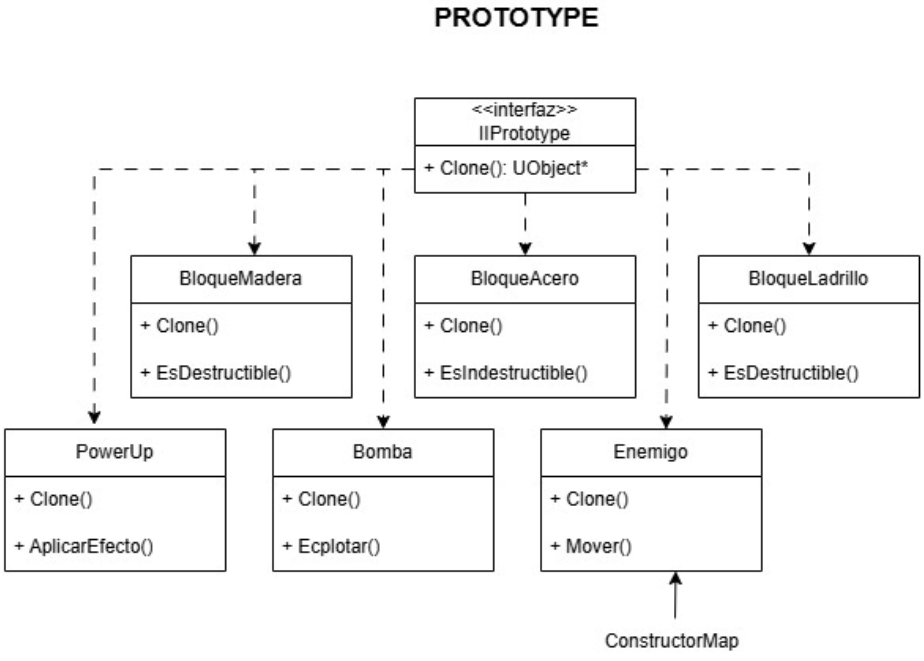
- Observer: Define una dependencia uno-a-muchos entre objetos, donde cuando un objeto cambia, todos sus dependientes son notificados automáticamente.

Por ejemplo: *UGameEventManager - Subject* que notifica eventos del juego, *UUIManager - Observer* que reacciona a eventos para actualizar la UI, *UVFXManager - Observer* que reacciona a eventos para crear efectos visuales y *IIGameObserver - Interfaz* que define los métodos de notificación
Eventos notificados:

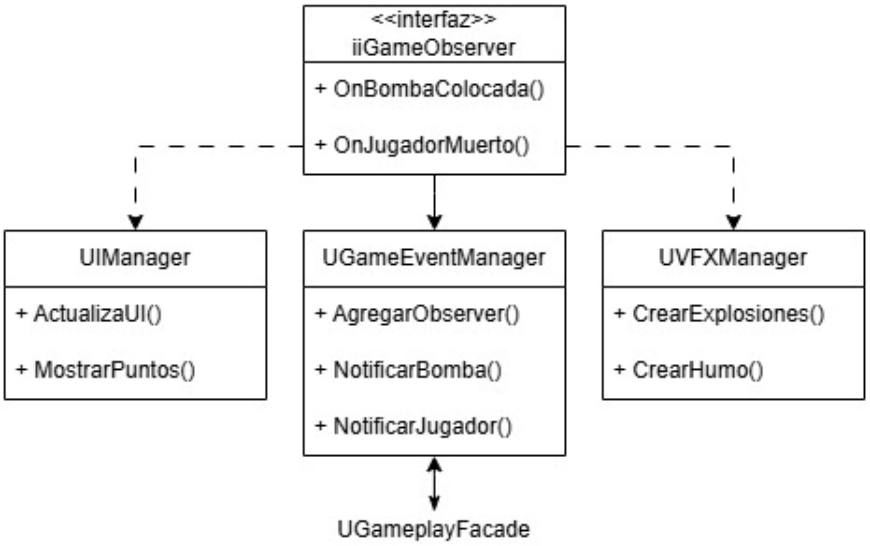
- Bomba colocada/explotada
- Jugador muerto/herido
- Power-up recogido
- Bloque destruido
- Enemigo eliminado
- Nivel completado

- Iterator: Facilita la exploración ordenada de celdas, enemigos, bloques o potenciadores sin exponer directamente la estructura interna. Es útil, para proporcionar una forma de acceder secuencialmente a los elementos de una colección. Por ejemplo: *UMapTileIterator - Iterator* para recorrer tiles del mapa y *IIIterator - Interfaz* que define los métodos del iterator
  Tipos de iteración:
  - Iteración general por todos los tiles
  - Iteración solo por tiles destructibles
  - Iteración solo por tiles vacíos
  - Iteración solo por tiles con bloques

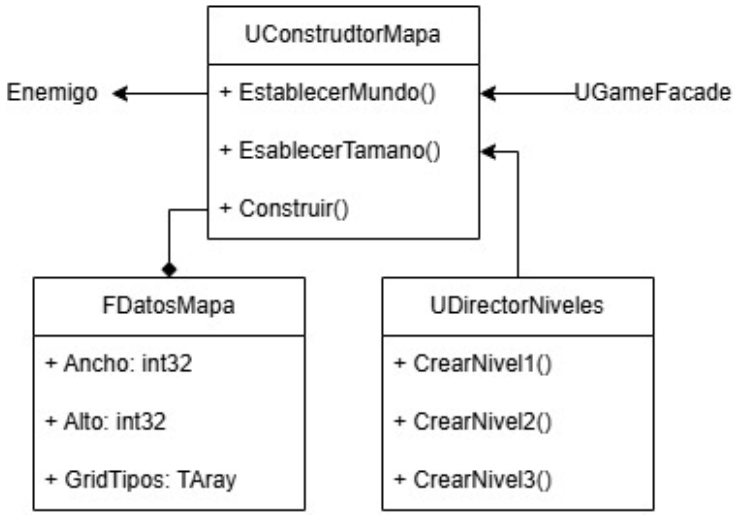
**2. Diagrama de Clases de los objetos y patrones**



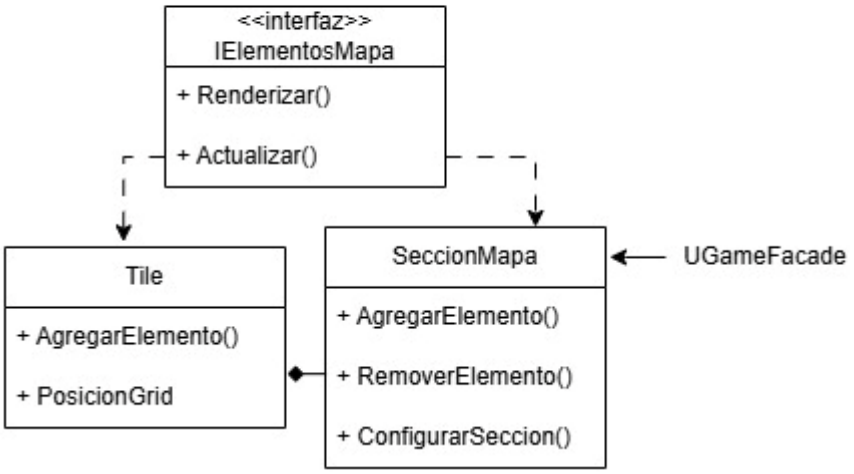
## OBSERVER



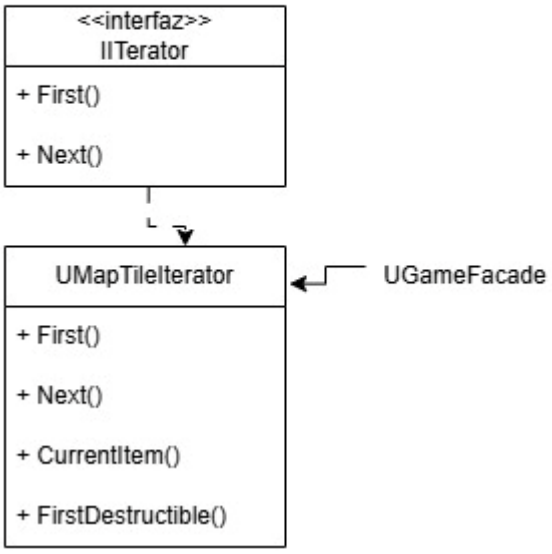

## BUILDER



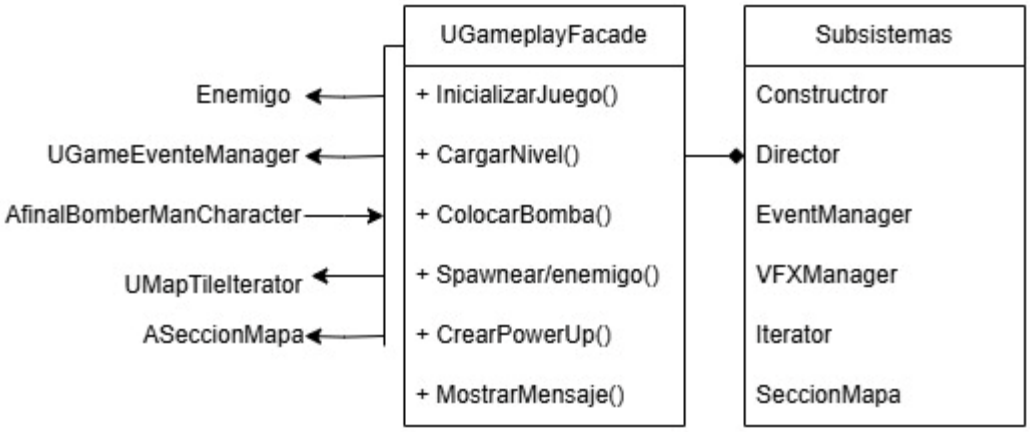
## COMPOSITE




## ITERATOR




## FACADE




Git

https://github.com/Sami-1320/Final_Bomberman