

Intel y Gentes

Documento de Diseño

Autores: Agüero, David - Bastida, Lucas - Miranda, Noelia - Palmiotti, Mauro

Fecha: 19/06/2020

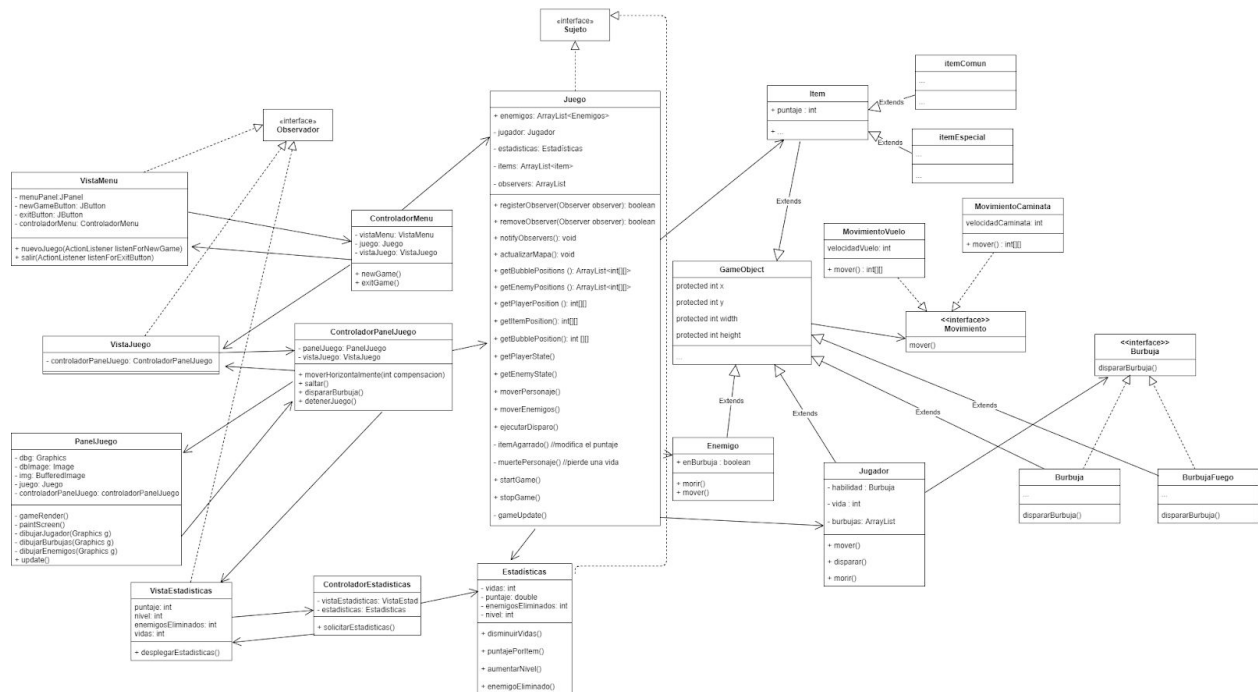
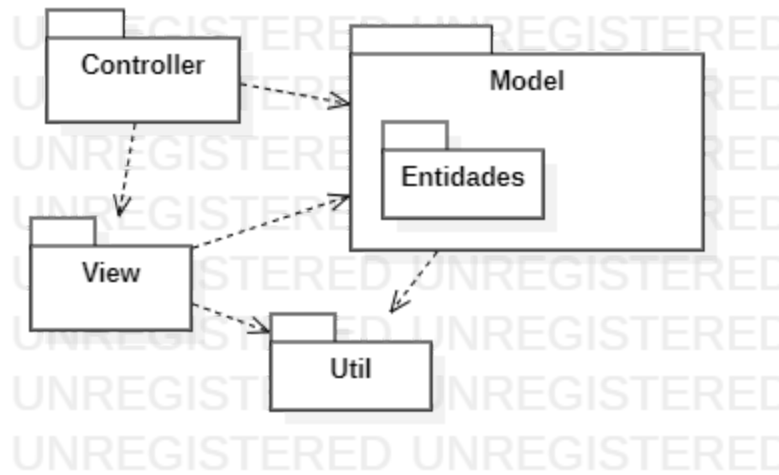
Version del Documento: 1.0.0

Historial de cambios

Versión	Fecha	Resumen de Cambios	Autor
0.1.0	14/06/2020	documento inicial	Intel y Gentes
0.2.0	18/06/2020	Actualización: diag y correcciones	Bastida, Lucas
0.3.0	18/06/2020	Diagrama de sec.	Miranda, Noelia. Palmiotti Mauro
1.0.0	19/06/2020	Actualización: Utilización de patrones de diseño	Aguero, David

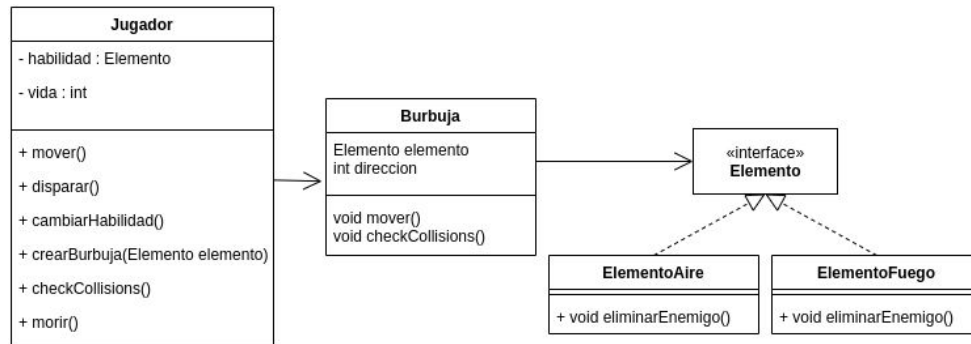
Diagrama de paquetes	3
Diagrama de clases	3
Utilización de patrones de diseño	4
Diagrama de secuencia	6

Diagrama de clases

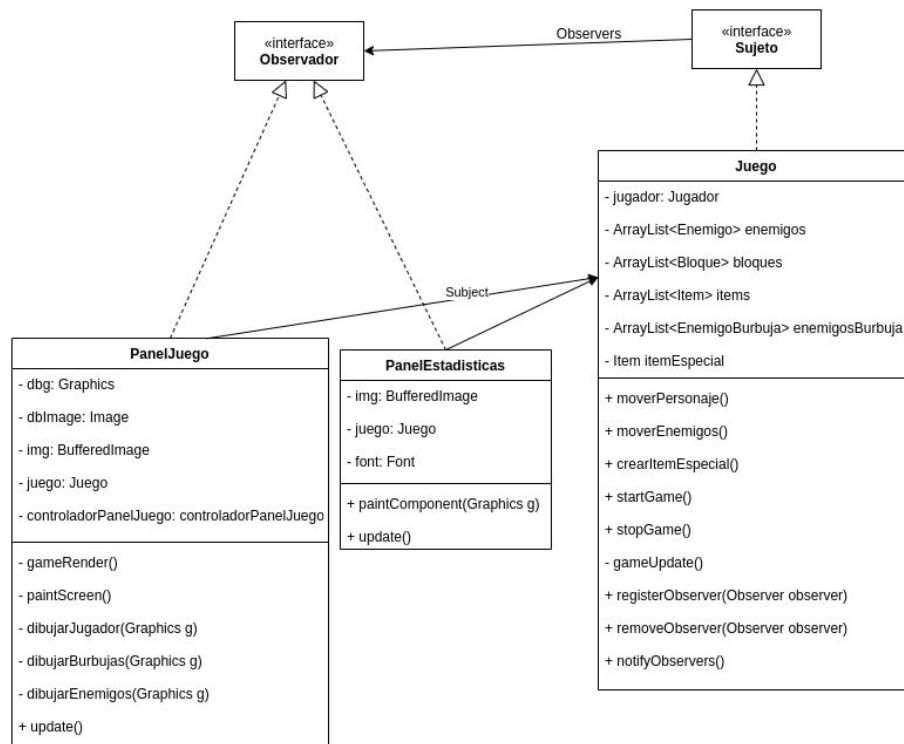


Utilización de patrones de diseño

Patron Strategy:



Patron Observer:



El patrón MVC que utilizamos para la arquitectura, es una composición de los patrones Observer y Strategy. Estos patrones, trabajan juntos para desacoplar las vistas del juego de la lógica del modelo, lo que permite mantener el diseño claro, flexible y extendible.

El modelo (la clase juego) contiene la lógica del funcionamiento del juego. Este modelo usa el patrón Observer para mantener a sus “observadores” (PanelJuego y PanelEstadísticas) actualizados y aún así desacoplados. Gracias a esto, en futuras versiones se podrían incorporar nuevas vistas al juego.

Las vistas y los controladores, implementan el patrón Strategy. Las vistas son objetos configurados con una estrategia, la cual es provista por los controladores.

Las vistas solo se preocupan por los aspectos visuales del juego y delegan a los controladores la responsabilidad de interactuar con el modelo para responder a los inputs del usuario.

Por otro lado, implementamos el patrón Strategy para poder intercambiar el tipo de burbuja que el personaje puede disparar. Esto nos permite realizar estos cambios en tiempo de ejecución, y también nos permite introducir modificaciones en posteriores versiones del juego.

Por ejemplo, cuando el jugador, recoge un ítem especial, la burbuja normal que puede disparar, es intercambiada por una de fuego.

El patrón Observer, lo utilizamos además para mantener una clase de estadísticas actualizada con los eventos que van aconteciendo durante la partida, tales como pérdida de vidas, cantidad de enemigos eliminados, puntaje, etc.

```

sequenceDiagram
    actor Actor1 as Lifeline1: Actor1
    participant Launcher
    participant Sec as Sec
    participant ControladorMenu
    participant VistaMenu
    participant Juego
    participant PanelJuego
    participant VistaJuego
    participant ControladorPanelJuego
    participant Jugador
    participant Burbuja
    participant Enemigo
    participant VistaEstatistica

    Actor1->>Launcher: 1: new Launcher
    activate Launcher
    Launcher->>ControladorMenu: 2: new ControladorMenu()
    deactivate Launcher
    activate ControladorMenu
    ControladorMenu->>VistaMenu: 3: new VistaMenu()
    deactivate ControladorMenu
    activate VistaMenu
    VistaMenu->>Juego: 4: new Juego()
    deactivate VistaMenu
    activate Juego
    Juego->>PanelJuego: 7: new PanelJuego(juego)
    deactivate Juego
    activate PanelJuego
    PanelJuego->>VistaJuego: 5: new Jugador
    deactivate PanelJuego
    activate VistaJuego
    VistaJuego->>ControladorPanelJuego: 6: new Enemigo
    deactivate VistaJuego
    activate ControladorPanelJuego
    ControladorPanelJuego->>Burbuja: 
    deactivate ControladorPanelJuego
    activate Burbuja
    Burbuja->>Enemigo: 
    deactivate Burbuja
    activate Enemigo
    Enemigo->>VistaEstatistica: 
    deactivate Enemigo
    activate VistaEstatistica
    VistaEstatistica->>ControladorPanelJuego: 8: new VistaEstatistica(juego)
    deactivate VistaEstatistica
    activate ControladorPanelJuego
    ControladorPanelJuego->>PanelJuego: 9: registerObserver(panelJuego)
    deactivate ControladorPanelJuego
    activate PanelJuego
    PanelJuego->>VistaJuego: 10: registerObserver(VistaEstatistica)
    deactivate PanelJuego
    activate VistaJuego
    VistaJuego->>PanelJuego: 11: new VistaJuego(panelJuego)
    deactivate VistaJuego
    activate PanelJuego
    PanelJuego->>ControladorPanelJuego: 12: ControladorPanelJuego(panelJuego, juego)
    deactivate PanelJuego
    activate ControladorPanelJuego
    ControladorPanelJuego->>Jugador: 13: start game
    deactivate ControladorPanelJuego
    activate Jugador
    Jugador->>ControladorPanelJuego: seq loop
    deactivate Jugador
    activate ControladorPanelJuego
    ControladorPanelJuego->>PanelJuego: 14: gameUpdate
    deactivate ControladorPanelJuego
    activate PanelJuego
    PanelJuego->>Jugador: 15: mover
    deactivate PanelJuego
    activate Jugador
    Jugador->>ControladorPanelJuego: 16: moverBurbuja
    deactivate Jugador
    activate ControladorPanelJuego
    ControladorPanelJuego->>Burbuja: 17: mover
    deactivate ControladorPanelJuego
    activate Burbuja
    Burbuja->>Enemigo: 19: checkCollision
    deactivate Burbuja
    activate Enemigo
    Enemigo->>ControladorPanelJuego: 20: moverEnemigo
    deactivate Enemigo
    activate ControladorPanelJuego
    ControladorPanelJuego->>PanelJuego: 21: mover
    deactivate ControladorPanelJuego
    activate PanelJuego
    PanelJuego->>VistaJuego: 22: notifyObserver
    deactivate PanelJuego
    activate VistaJuego
    VistaJuego->>PanelJuego: 23: update
    deactivate VistaJuego
    activate PanelJuego
    PanelJuego->>ControladorPanelJuego: 24: update
    deactivate PanelJuego
    activate ControladorPanelJuego
    ControladorPanelJuego->>PanelJuego: 25: paintscreen
    deactivate ControladorPanelJuego
    activate PanelJuego
    PanelJuego-->>ControladorPanelJuego: 
    deactivate PanelJuego
    deactivate ControladorPanelJuego
    deactivate Jugador
    deactivate Burbuja
    deactivate Enemigo
    deactivate VistaEstatistica

```