22/03/2025

Alexandru Darie
Aitor Cotano
Diego Pomares
Ivan Andrés
Peio López

Informe: Patrón MVC + Observer

El patrón MVC (Modelo-Vista-Controlador) es una forma de estructurar aplicaciones separando claramente la lógica, la presentación y la interacción del usuario. En este enfoque, el **Modelo** representa los datos y la lógica del sistema. La **Vista** es la interfaz gráfica encargada de renderizar esos datos. Finalmente, el **Controlador** es quien recibe las acciones del usuario (input) y comunica esas acciones al modelo para que este actúe en consecuencia.

Por otro lado, el patrón Observer permite que los observables (que estan dentro del modelo) notifiquen automáticamente a la todos los observadores (la vista) cuando sus estados cambian. De esta forma, la vista puede actualizarse sin que el modelo tenga una referencia directa a ella. Este desacoplamiento es una de las grandes ventajas del uso conjunto de ambos patrones.

Controlador, Modelo y Vista:

El **controlador** cumple el rol de intermediario entre la entrada del usuario y el modelo. En este caso, está implementado como un KeyListener que detecta cuándo el usuario presiona una tecla y llama a los métodos correspondientes del modelo, como mover al personaje o soltar una bomba. El controlador no interactúa directamente con la vista, ni maneja la lógica de presentación. Su única responsabilidad es traducir las acciones del usuario en instrucciones para el modelo.

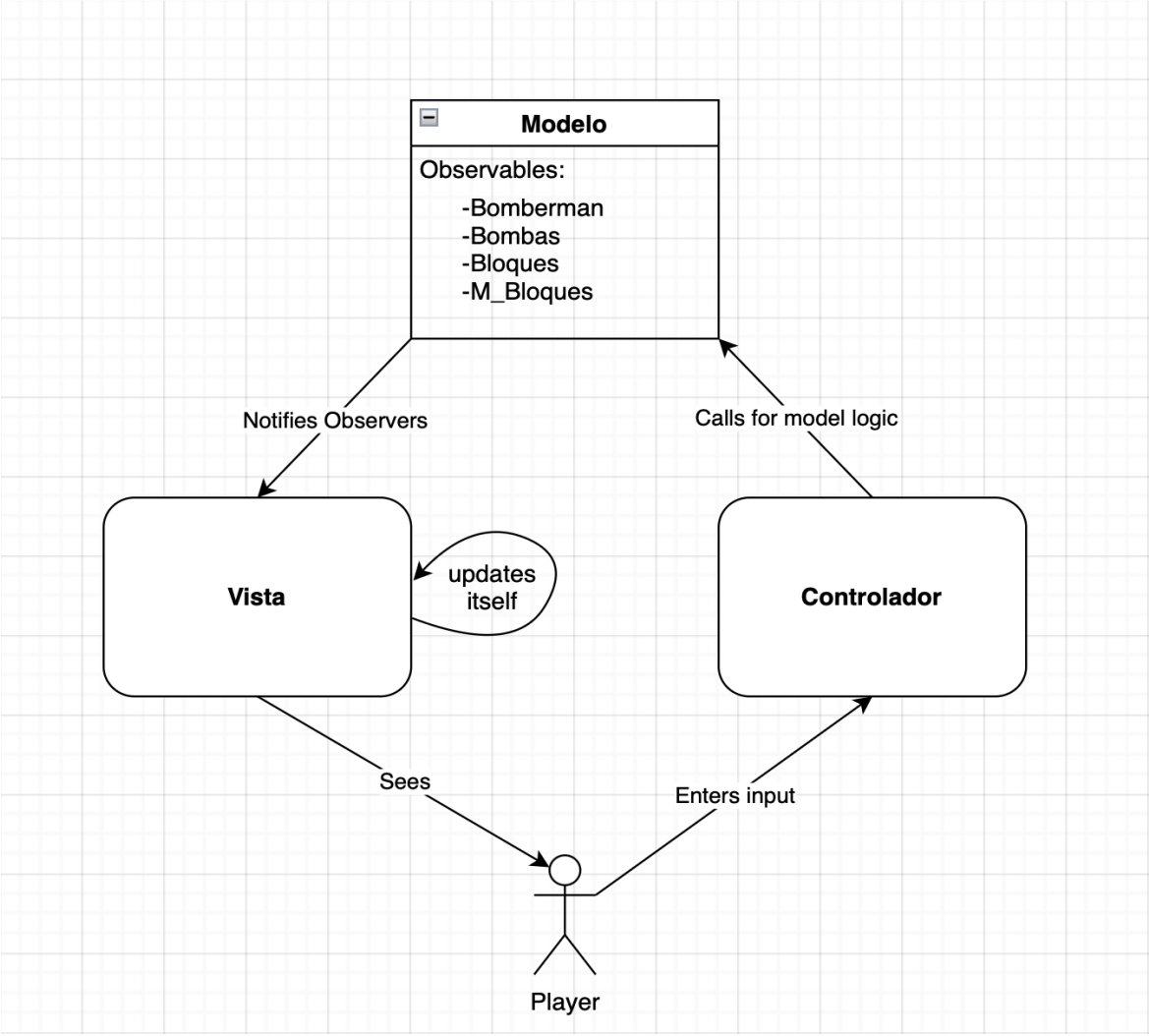
Por otro lado, tenemos al **modelo** Cada vez que el estado de uno de sus componentes (Bomberman, Bomba, etc..) cambia, dicho elemento notifica a todos la vista. Por ejemplo, un bloque es destruido en la logica. El bloque notifica que ha sido destruido a la vista.

En este caso, la **vista** está registrada como observador. Se actualiza automáticamente acorde con las notificaciones recibidas de los observables. La vista interpreta los argumentos de las notificaciones para solo realizar las actualizaciones necesarias. Por

ejemplo, se ha recibido una notificación que indica que un bloque ha sido destruido. Se actualiza el sprite del bloque especificado.

Notify y Update:

Ya se ha descrito la relación entre el modelo y la vista. El modelo notifica a la vista de los cambios que hay que hacer, y esta se actualiza automáticamente. En este caso, la mayoría de elementos del modelo al ser un videojuego son observables mientras que la vista cumple el rol de observador. Los elementos observables utilizan un método llamado *notifyObservers()* el cual recibe argumentos de tipo *String* para indicar el tipo de cambio que se va a llevar a cabo. El método *Update()* en la vista utiliza estos argumentos para realizar el cambio pedido. Por ejemplo, Bomberman muere. En este caso, se notificará a la vista con el siguiente argumento: *"Dead"*. El método U*pdate()* de la vista, al leer el argumento, llama al método privado *actualizarMuerte().*



Este flujo de comunicación permite que el modelo y la vista estén completamente desacoplados. El modelo no necesita saber qué vista lo observa, ni cómo se actualiza visualmente. Solo se encarga de mantener el estado del juego y notificar cambios. La vista, al recibir esos eventos, decide cómo representarlos gráficamente. Este mecanismo hace que el sistema sea más mantenible, modular y flexible.