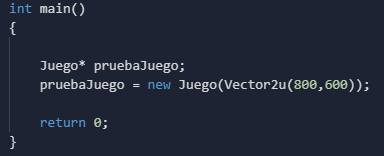
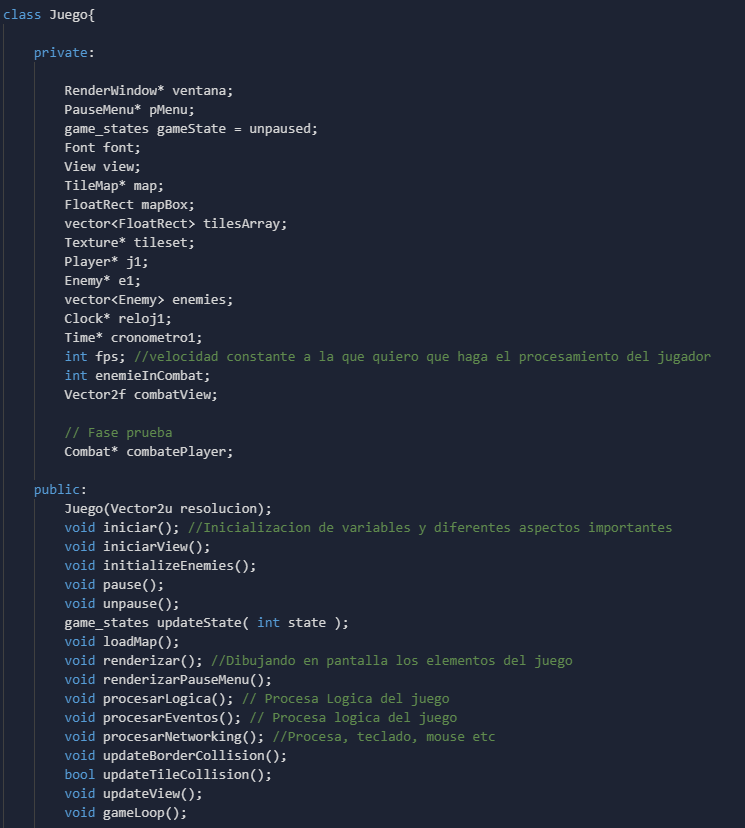
**Manual Tecnico**

Principales funcionalidades:

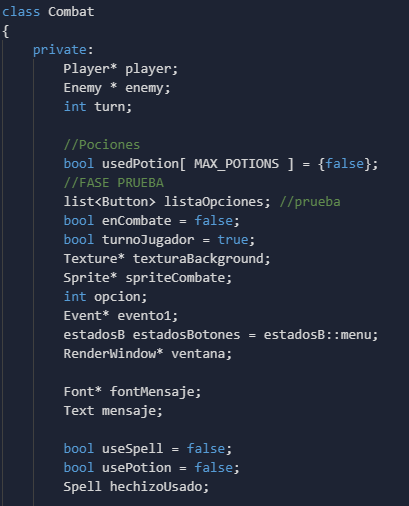


Empezando por el main, que es el encargado de llamar a la clase Juego para crear el objeto pruebaJuego que se encarga de inicializar la pantalla y dentro se encarga de inicializar funciones que son vitales para el correcto funcionamiento del videojuego



Dentro de la clase juego tenemos 16 funciones privadas relevantes, ya que todas ellas se encargan de suplir una necesidad para el entorno gráfico o el correcto funcionamiento de este. Entre sus funciones públicas se encuentran principalmente inicializadores, de este apartado me gustaría destacar procesarLogica, procesarEventos y procesarNetworking; encargadas de la lógica, eventos y periféricos, respectivamente.

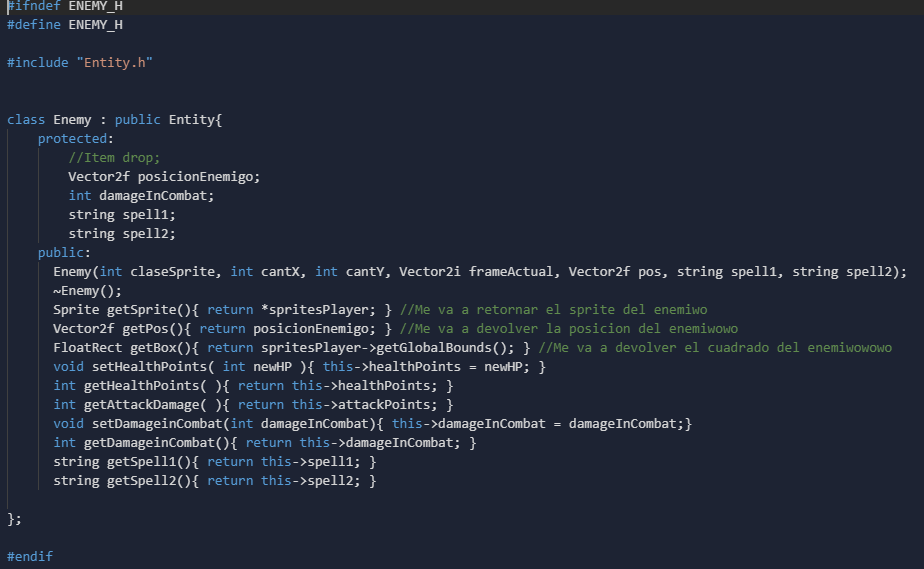
En el apartado de combate todo es relevante para el desarrollo del mismo; pero me gustaría destacar la función RenderWindow que es la encargada de quitar la mazmorra para poner el escenario donde ambos sprites se encuentran frente a frente y se muestran los botones para la selección de habilidades del jugador



En las funciones públicas de la clase combate nos encontramos con un constructor que se encarga de crear la nueva ventana donde se mostrarán los botones, enemigos, al jugador y a los respectivos ataques de cada uno. Además de que aquí se incluyen las funciones de dibujado y renderizado de todo lo que se verá en pantalla, tales como los paneles de opciones, el inventario, las habilidades y los mensajes de cuando se aplica y recibe daño.



Clase encargada de la inicialización del enemigo, junto con su sprite, su nombre, su vida. etc.



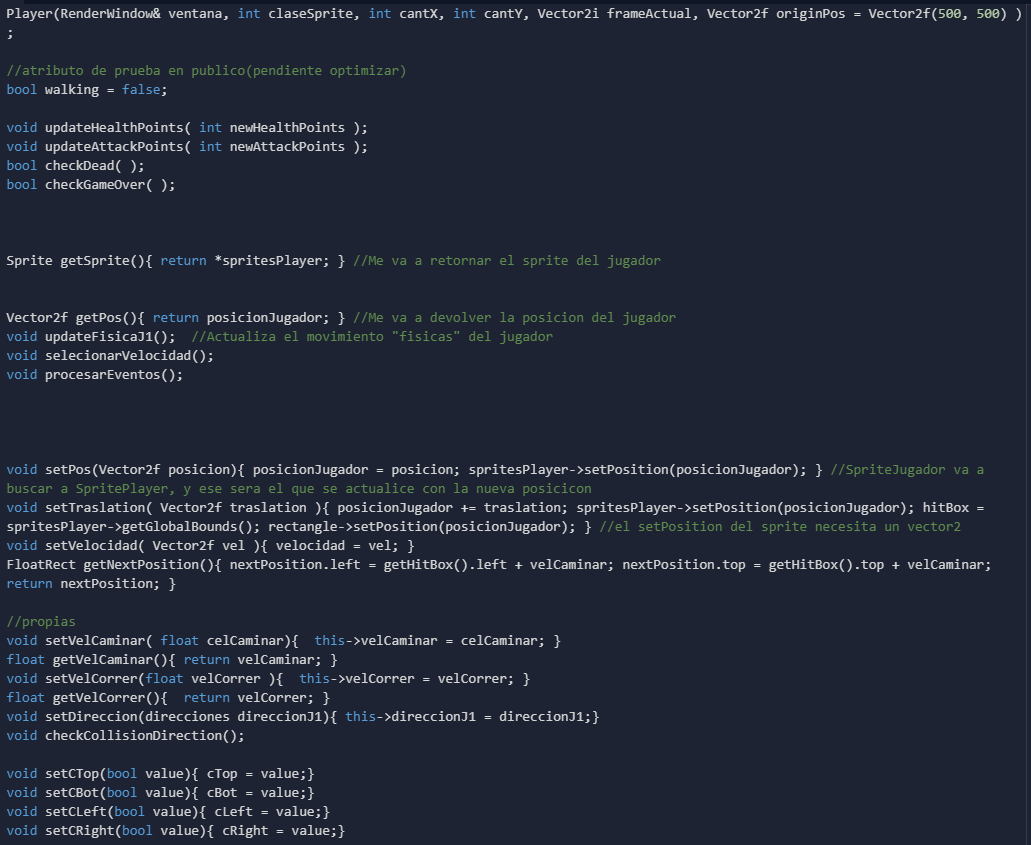
La clase entidad se encarga de aplicar las bonificaciones que tendrá cada item (espada, pechera, perneras, pociones. etc.



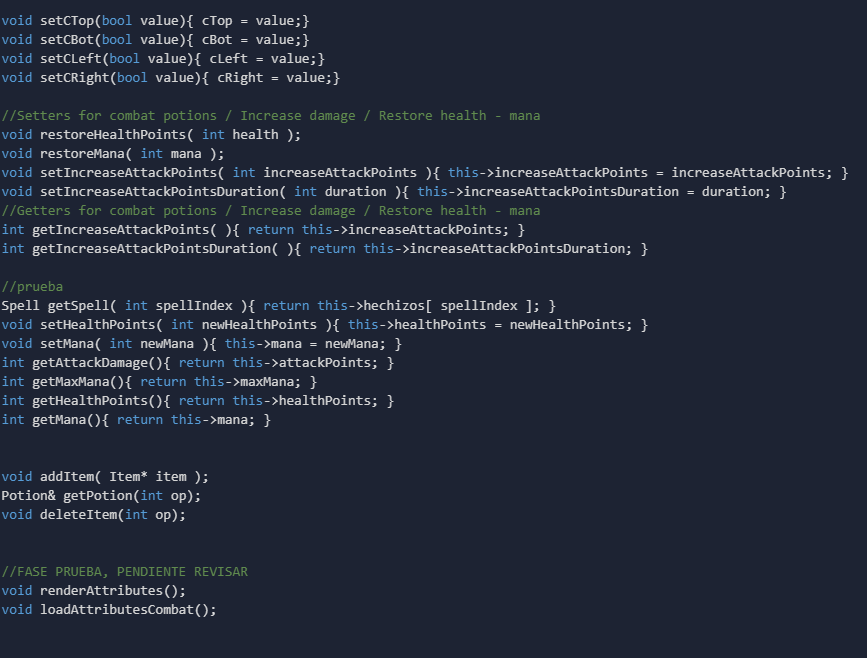
La clase Player, la cual es la más grande con diferencia, ya que el juego gira en torno a él.



En su apartado privado tenemos los incrementos que se realizan en sus atributos, tales como puntos de ataque y durabilidad, sus cantidades de maná, corazones y xp; su velocidad al caminar, la dirección a la que mira y las interacciones que tiene el teclado con él. Además de sus texturas y un vector donde se guardan sus habilidades (spells).

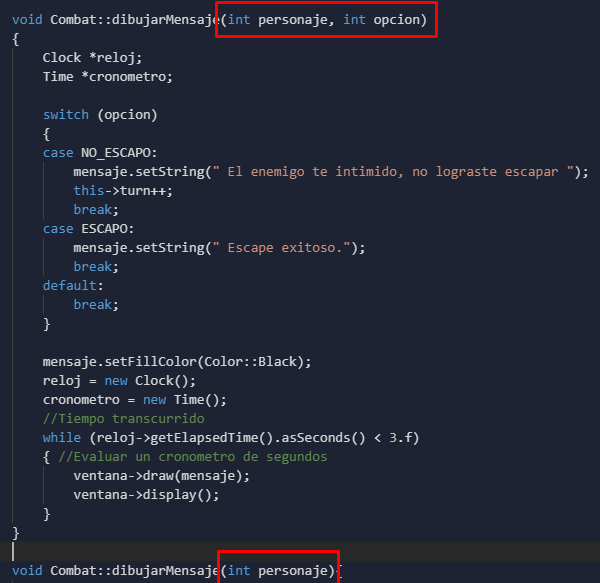


En el apartado privado de la clase Player se encuentran las actualizaciones que se realizan a sus atributos tales como la vida y los puntos de ataque, también está el verificador de si el jugador se encuentra vivo o muerto, al igual que el verificador de si el juego continúa o no, funcionalidades relacionadas con sprites tales como setTraslation que se encarga de mover el sprite para dar el efecto de animación.

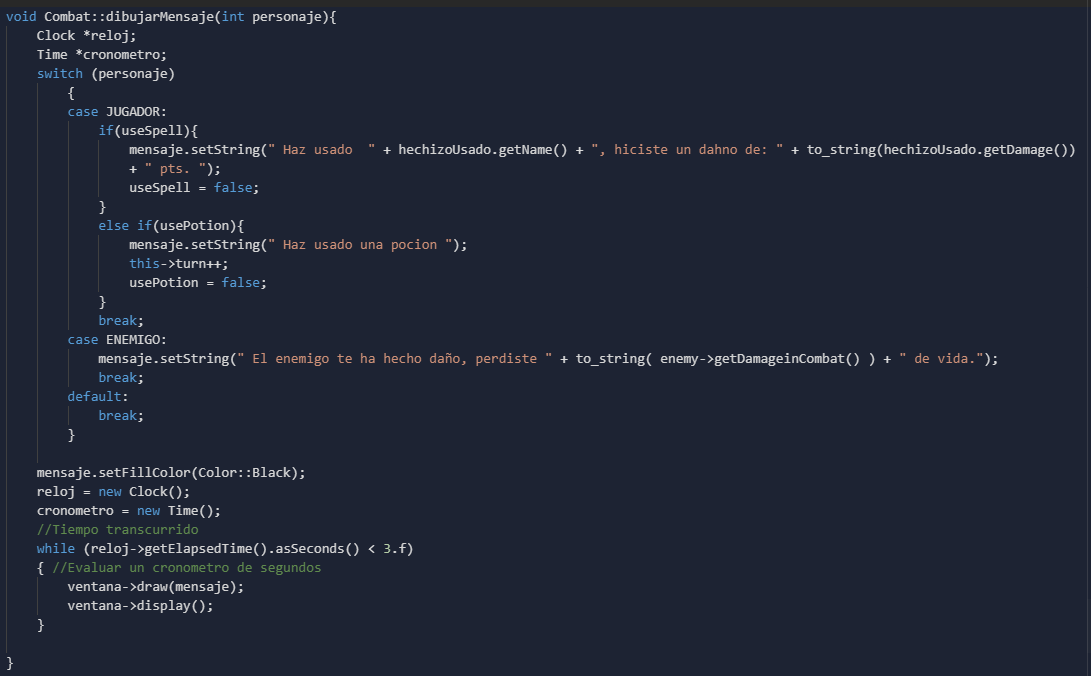


En el mismo apartado se encuentran funcionalidades de tipo Setter y Getter encargadas de poner y brindar información respectivamente; todo esto basándose en atributos privados de la clase Player.

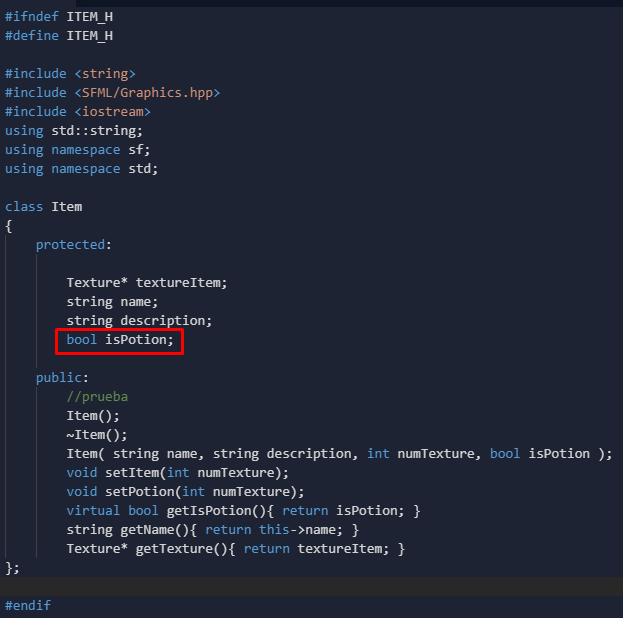
**Sobrecarga**



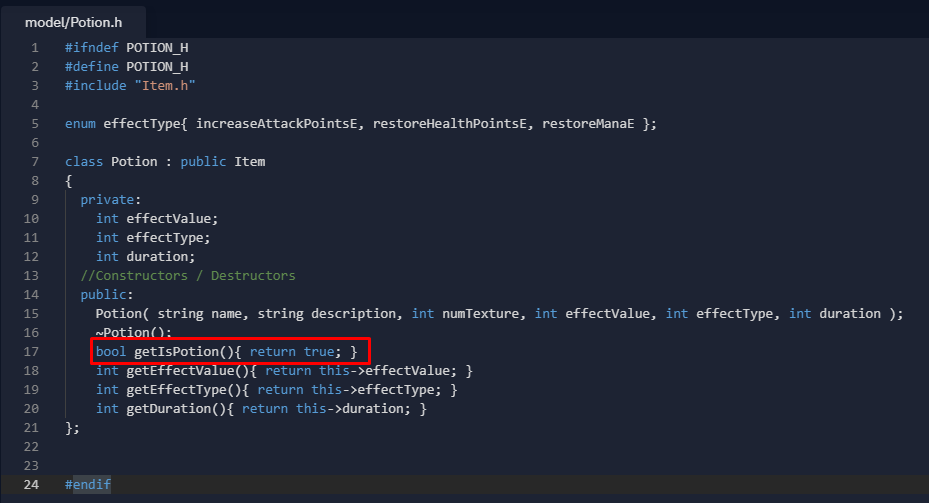
En la función dibujarMensaje podemos evidenciar cómo utilizamos la misma función, pero con un parámetro extra en el caso del uso que se le da en la primera función, donde se utiliza opcion para definir el mensaje que se mostrará, a diferencia del segundo uso que se le da, donde simplemente se le usa para imprimir mensajes.



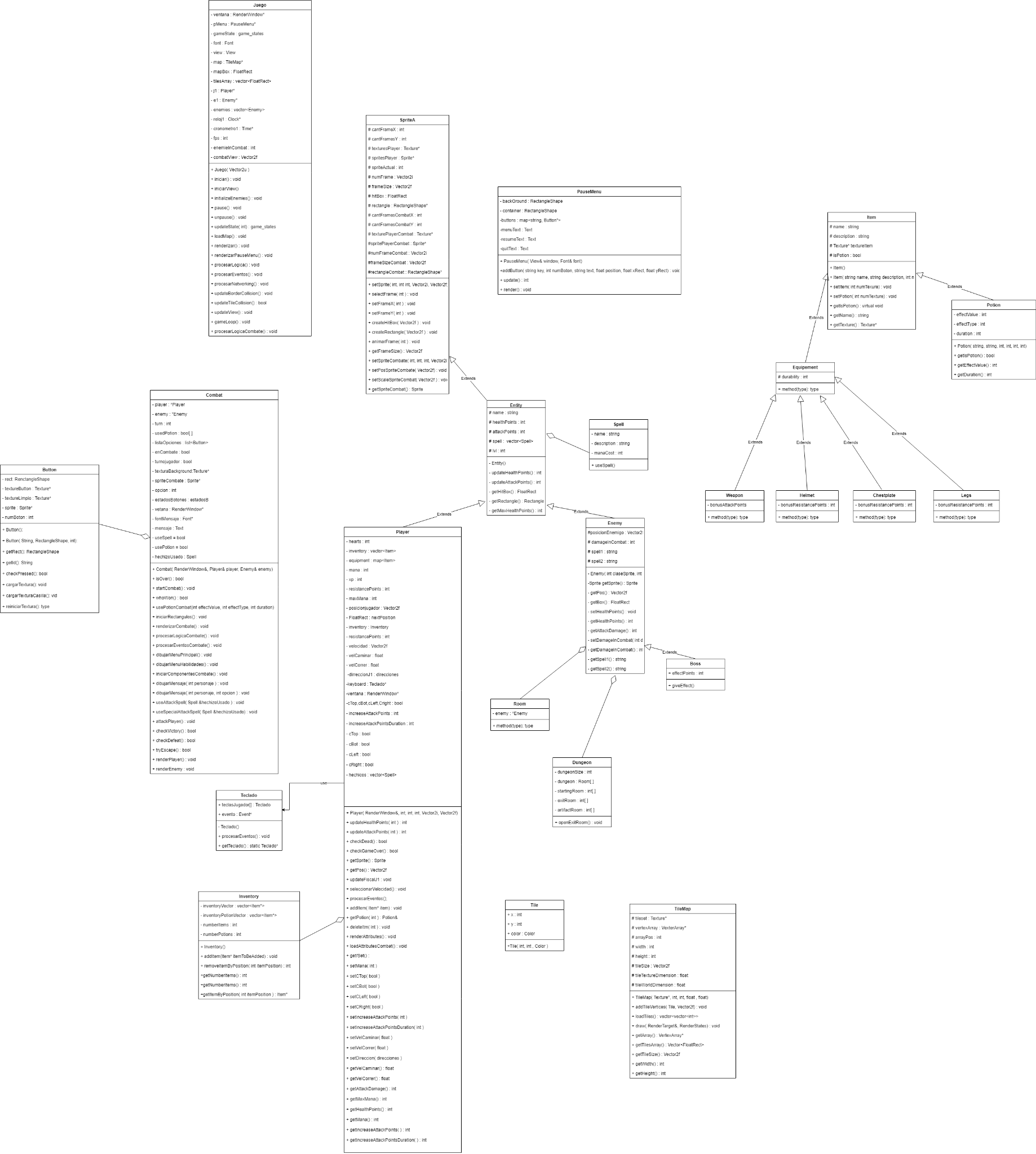
**Clase Abstracta y Polimorfismo**



En la clase item tenemos el booleano isPotion, que nos define si un item es una poción o no, contamos con la función virtual getIsPotion que nos retorna este valor booleano para usarse en la clase Potion



**Gráfico UML**

****

Para una mejor visualización: https://app.diagrams.net/#G1YyJkyx7dDGmc8kjPmDLyDfibYYyriZab