Interacción del jugador con el mapa

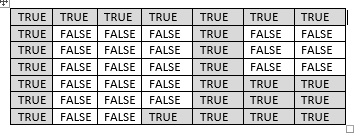
Para explicar como el jugador interactua con el mapa y los diferentes elementos, primero es necesario explicar las dos clase fundamentales sobre las que se crea la mecánica del juego, más allá de los combates, menús y personajes del mismo.

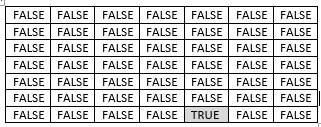
La primera clase fundamental es “EstadoMapaJuego”, está clase que hereda de “BasicGameState” es la encarga de renderizar, actualizar y controlar las acciones del jugador en el mapa. Para ello cuenta con un objeto de clase “TiledMap”, del cual se extraen todos los datos y atributos necesarios para el control de eventos y colisiones con el mapa mediante código.

Al crear dicho objeto, denominado “map” de aquí en adelante, se han diseñado 3 funciones principales para obtener el mapeado de todos los atributos de tipo booleano grabados en el mapa. Para hacer este “mapeado” de atributos mediante código, se implementan varias matrices, una por cada atributo implementado, de tamaño equivalente al numero de “tileds” u cuadrados que posee el mapa cargado en el momento, haciendo que cada posición de la matriz equivalga al valor del atributo que trata en la misma posición del cuadrado del mapa.

Esto quiere decir que sabiendo en cada momento sobre que cuadrado se encuentra el jugador, para averiguar si hay algún tipo de evento en ese lugar solo debemos revisar dicha posición sobre las matrices de atributos.

Ejemplo: Mapa 7x7 y ejemplo de dos de sus matrices de control de atributos.

 Array de control del atributo “bloqueo”

Array de control del atributo “NPC”

Para controla al posición del jugador, así como aquello con lo que choca y poder hacer la comprobación correspondiente, entra en juego la segunda clase: “Heroe”. Esta clase contiene el código que permite realizar tres tareas de gran importancia.

En primer lugar utilizando como dato de entrada la tecla de dirección que se este pulsando en ese momento o la última pulsada, si no se esta pulsando ninguna, actualiza como debe renderizar al heroe en la pantalla, ya sea quieto mirando hacia un lado o moviendose en una dirección mientras la animación se ejecuta fotograma a fotograma.

En segundo lugar contiene su propia posición dentro de un vector bidimensional, dicho vector contiene la posición que guarda con respecto al mapa en el que se encuentra en píxeles, es decir nos proporciona la información donde debermos dibujar a nuestro jugador

Además de ello, conociendo esta posición píxelica y puesto que conocemos perfectamente el ancho y alto de los cuadrados de nuestra cuadricula, haciendo una simple transformación podemos conocer sobre que cuadro del mapa estamos dando lugar a la tercera tarea importante que realiza esta clase: el control de colisiones.

Detección de colisiones y control de eventos

Como hemos mencionado el control de colisiones se lleva a cabo dentro de la clase “heroe”, pero para ello es necesario que cuente con los arrays de control de atributos propuestos dentro de la clase “EstadoMapaJuego”, por ello ambas clases deben guardar algun tipo de realción. En nuestro caso nos hemos decantado por el uso de clases anidadas, es decir crearemos un único objeto de la clase “EstadoMapaJuego” en cuyo interior se creará un atributo que se corresponda con el un objeto de la clase “heroe”.

Con esto echo contamos con la posibilidad de operar todas las funciones de ambas clases desde un único bucle de control del juego.

La colisión con estos atributos se hace relacionando la trayectoria del avatar del jugador con la casillas que contienen los atributos. Para esto utilizamos como datos de entrada los botones de dirección que sirven como mandos del control de movimeinto, la posicón que tenga el jugador en un momento determinado y por último la casilla sobre la que se prevee que el jugador se encontrará atendiendo a los parámetros anteriores. Tras encontrar la casilla sobre la que se prevee que el jugador estará siguiendo esa dirección solo debemos analizar su posición en los diferentes arrays para encontrar si habrá o no alguún efecto de respuesta.

Un buen ejemplo de esto sería un jugador pulsando la flecha de dirección hacia la derecha que se encuentra en la casilla (10,15). Atendiendo a estos dos factores se puede predecir que su posición en el eje X aumentará y su siguiente paso lo situará sobre la casilla (11,15). Con esta información, si quisieramos comprobar si la siguiente casilla es adsequible solo deberíamos comprobar nuestro array “blocked” en la posición (11,15), si el valor de dicha posición es “true” entonces la casilla es inadsequible y no debemos dejar al avatar actualizar su posición en esa dirección.

De esta forma contaremos pues con los siguientes arrya de atributos para cada mapa y cada colisión con alguno de ellos provocará un efecto u otro:

-Casilla con atributo“Blocked”: cualquier casilla que posea este atributo será inacsequible por el jugador, es decir l se encontrará bloqueada y por tanto la figura del personaje será incpaz de situarse sobre esas casillas. Corresponden a estas, las casillas de objetos grandes, NPCS o paredes sobre las que uno no se puede situar.

Al chocar con las casillas que posean este atributo con un valor “false” el valor de la posición del jugador se actualizará en función de la dirección que se este pulsando en ese momento, determinando su nueva posición en los ejes X e Y.

Si por el contrario el atributo posee un valor “true” la actualización de posición no se llevará a cabo y el jugador permanecerá en el sitio contra la casilla bloqueada, dando la sensación de “choque” o colisión contra el objeto.

-Casilla con atributo “enemigos”: estas casillas controlan la aparición de combates aleatorios en el mapa. La colisión del jugador con cualquier casilla con este atributo como “true” provocará la generación de un número aleatorio comprendido entre 0 y 1, el valor producido será comparado con un valor de tasa de aparición, y en caso de que entre dentro de unos valores establecidos se saltará de estado de juego hacia un combate con generación de enemigos aleatorio en función del mapa.

-Casillas con atributo “evento”: la colisión con estas casillas pueden provocar diversos efectos, tantos como distinto eventos existen en el juego:

-Eventos de dialogo con NPCS: La colisión con estos eventos hara que se renderice un pequeño bocadillo junto al NPC con el que se ha chocado, permitiendo leer el dialogo del mismo y proporcionando información sobre el mundo al jugador. Para cerra dicho dialogo se debe utilizar la tecla “Enter” que solo hace efecto cuando se ha activado uno de estos eventos

-Eventos de Tienda: estos eventos son, como su nombre indica, el salto hacia el estado de juego Tienda, sobre el cual se cargará un vendedor con una serie de items u otros en función del mapa en el que se encuentre. Como se explica en el apartado sobre el estado de juego “Tienda” el jugador puede salir manualmente mediante el enu que se le presenta.

-Eventos de escenas: esto eventos más especiales, solo se activarán una vez al colisionar con una de las casillas del grupo que las activa. Esto es debido a que este tipo de evento realiz un salto sobre el estado de juego hacia uno de los estados que renderizan las escenas animadas que hacen avanzar la historia. Tras la escena esta se ocupará de redirigir al jugador hacia el siguiente combate o mapa de forma dnámica sin que el jugador deba hacer nada más que disfrutar con el avance de las historia. Estetipo de evento tambien es utilizado para el cambio de mapa.

Eventos de “Healer” o curandero: este evento conjunta el efecto de los eventos de dialogo pero añade el la propiedad de recuperar la vida y maná de todos los personajes del grupo permitiendo la resurrección de aliados caido.

Puesto que el detector de colisiones utilizado solo permite captar parámetros booleanos del mapa diseñado en Tiled Maps, se debe realizar la distinción entre los distinos eventos mediante código, esto se realiza mediante coordenadas de mapa.

Cada vez que se detecta la colisión con un “evento”, se capta la coordenada de la casilla que contiene el atributo “eventos=true”y se redirecciona sus coordenadas hacia la clase “EventosNPC” para su busqueda en una base de datos. Sobre esta clase se ha diseñado algoritmo de control de eventos en función del mapa y sus coordenadas. Las funciones de la clase analizan la posición del evento y aportan a la clase “heroe” todos los parámetros que necesita para saber de que tipo son y como abordarlos, consguiendo así que un único atributo booleano permita activar 4 tipos de eventos diferentes.

La tienda

Como en todo buen RPG se necesita una forma de conseguir objetos y negociar con ellos, esto se hace mediante el uso de la clase “EstadoTienda” o “Tienda” para abreviar. Dicha clase hereda sus propiedades de la clase “BasicGameState” por lo que cuenta tanto con un bucle de control como con un bucle de rendeizado.

Como se explica en apartado correspondiente este estado se pone en acción cuando el jugador colisiona contra un evento de tipo de “Tienda”, en el cual se genera un objeto de tipo vendedor que posee un listado items sobre el que poder negociar.

La tienda cuenta con una interfaz sencilla y una mecánica basada en un flujograma de funcionamiento simple.

IMAGEN DEL FLUJOGRAMA DE TIENDA

La interfaz gráfica de la tienda es bastante simple y renderiza una serie de menús en función del punto del flujograma en el que se encuentre el jugador, además de la ayudante del vendedor “Anna” que siempre estará dipuesta comentar las características de los objetos y decirte el dinero que te queda.

El menú principal en el que el jugador se encontrará al entrar poseerá 3 opciones simples: Comprar, Vender y Salir.



Saltandonos por un instante las dos primeras opciones, “Salir” devolverá al jugador al mapa en el que se encontraba y en la misma posición, es decir saltará del estado “Tienda” al estado “MapaJuego”.

Utilizando la opción “Comprar” el menú cambiará, y en esta ocasión se listarán los objetos que el vendedor posee en ese intante, junto con su coste y la descripción del objeto seleccionado en el instante por e jugador. En caso de que el objeto a veder sea un consumible tambien se mostrará la cantida restante que puede vender el comerciante antesde quedarse sin existencias.

Cuando el jugador utiliza el botón “Enter” para confirmar el objeto el programa comproborá el precio del objeto y lo comparará con la cantidad que posee el inventario del jugador, después comprobará si hay espacio en el inventario o dentro de su apartado de consumibles, una vez realizadas las comprobaciones necesarias, si el objeto puede ser comprado con éxito se eliminará de la lista de items en venta y se añadirá al inventario del jugador, descontando el dinero correspondiente.

Por otro lado si utilizamos la opción “Vender” el proceso es el contrario y no requiere de tantas comprobaciones. Se listarán los objetos presentes en el inventario del jugador junto con su precio de venta y la descripción del objeto seleccionado en el momento. Al seleccionar el objeto mediante “Enter” el objeto desaparecerá del inventario y se agregará el dinero de la venta.

Por añadir cierta dificultad al juego el objeto vendido no va a parar al inventario del comerciante, si no que este desaparece para siempre.

