

#### Punto 0:

En este punto, se nos explica cómo vamos a crear una clase estado de la que heredarán nuestros estados, que contendrá cuales van a ser esos estados y los eventos necesarios de todo estado, Enter, Update, y Exit; y una serie de métodos y procesos que indican cual de estos debe ejecutarse dentro del propio estado.

#### Punto 1:

En este punto, vamos a crear una clase singleton llamada GameEnvironment, que contendrá una lista de los checkpoints que se usarán en el estado de patrulla, con un método que creará una instancia donde se buscará esos checkpoints por su tag, se añadirá a la lista y por último se nos devolverá la lista con los checkpoints.

A continuación, se creará la clase State, a la cual añadiremos la librería UnityEngine.AI para poder utilizar el componente NavMesh. Esta clase contendrá una Enum con los estados que vamos a definir más adelante, seguido de otra Enum para los eventos dentro de cada estado, Enter, Update y Exit. Tendremos también una serie de variables, entre las cuales tenemos un Estado como el que hemos definido en el primer Enum, que contendrá el nombre del estado actual, y otro que hará referencia al siguiente estado al que se va a cambiar, además de otras variables que serán utilizadas dentro de los estados. Después, crearemos un constructor para los estados, donde enlazamos las variables definidas previamente que estos utilizarán en sus procesos. A continuación definiremos las transiciones entre eventos, y un método llamado Process donde se realizarán estas transiciones, y se devolverá cual es el siguiente estado al que cambiar una vez se ejecute el evento Exit, o el estado en el actual en caso contrario.

#### Punto 2:

En este punto, vamos a crear los estados Idle y Patrol, los cuales heredaron de la clase Estado que hemos creado anteriormente. Para ello, primero tendremos que definir el constructor de la clase en cada estado, donde también debemos asignar el nombre del estado. Después, pasamos a definir los tres métodos para cada evento, Enter, Update, y Exit.

En Idle, dentro de Enter, iniciamos en el componente animador la animación correspondiente antes de pasar al update, en Update, crearemos una condición que rompa el loop del update, dentro de la cual definiremos cuál será el siguiente estado al que se pasará después de llegar a Exit, y por último, en Exit, se reseteará el componente animador antes de transicionar al siguiente estado.

En Patrol, antes de definir el constructor, creamos una variable tipo int que se utilizará más adelante para navegar a través de la lista de checkpoints que nos proporciona la clase singleton GameEnvironment. Dentro del constructor, definiremos para el navmeshagent la velocidad para el movimiento del npc y que no esté parado, ya que queremos que durante este estado el npc se mueva acorde a un recorrido ya definido por los checkpoints. Una vez más, dentro de Enter iniciaremos la animación correspondiente, después, dentro del Update comprobaremos que el npc ha llegado a un checkpoint si la distancia entre el navmeshagent y el siguiente checkpoint es menor que 1, y después recorreremos la lista de checkpoints para saber cual es el siguiente checkpoint. Mientras el valor devuelto sea menor o igual que el valor máximo de la lista, asignaremos el siguiente checkpoint como su nueva destinación, de lo contrario, haremos que vuelva al checkpoint inicial para repetir el recorrido. Por último, en Exit, reseteamos el componente animador antes de pasar al siguiente estado.

Después, crearemos la clase IA, que controlará los cambios de estado y los cambios entre eventos dentro de estos. Añadimos la librería UnityEngine.IA para hacer uso del navmesh, y declaramos las variables necesarias, el navmeshagent, el animador, el transform del jugador, y una variable state que contendrá el estado actual en el que se encuentra el npc. Dentro del método Start obtenemos las referencias a los componentes de nuestras variables, he iniciamos la variable state en Idle. Por último, en el método Update llamamos al método Process, el cual se encargará de realizar los cambios de evento dentro de los estados y los propios cambios de estado a estado.