Search/Posing/Patrol

Para estos 3 estados he reutilizado esencialmente lo que tenía en la primera práctica, pero usando un MonoBehaviour para guardar los puntos de patrulla que hay en la escena y en el StateMachineBehaviour acceder a ello. Esto lo puede hacer gracias al video de contenido de unity. Le he añadido un IEnumerator para que tenga un Delay para cambiar su punto de patrulla y, al ser StateMachineBehaviour no puede usar el StartCorutine para activar la corutina. Entonces he investigado con StateMachineBehaviour + StartCoroutine() - Questions & Answers - Unity Discussions he aprendido a activar el IEnumerator del StateMachineBehaviour con el MonoBehaviour. También implementé un raycast para ver cuando llamar a un parametro de mi animator cambiando de estado, que ya sabía cómo funcionaban los parametros y las transicones del animator.

Hide

Para hacer el Thief se esconda dentro de un conducto, he optado a buscar todos los gameObject de la escena con NavMeshLink utilizando el FindObjectsByType aprendido en Unity - Scripting API: Object.FindObjectsByType (unity3d.com). Luego accedo al endPoint del componente de los NavMeshLink, convertiendolo de local space a world space usando TransformPoint()Unity - Scripting API: Transform.TransformPoint (unity3d.com). Busco el link que está más cerca el Thief con el for y Vector3.Distance y hacer el Thief se dirija hacia ello.

Para que se quede 5s quietos he procedido a crear un IEnumerator y que cuando termine la corutina que se vuelva a search.

Flee(Thief)

Para que el Thief huye del Guard he implementado resta y suma de vectores para saber hacia qué dirección sea la contraria entre el Guard y el Thief. De esta manera puedo calcular el destino de 10 metros que escapa del Guard. Para saber el Thief de qué Guard se escapa, hice que en la parte del search guarde la variable GameObject del Guard detectado. Luego con el GetBehaviour puedo acceder al otro StateMachineBehaviour y por consiguiente su variable público. Esto lo he aprendido de Unity - Scripting API: Animator.GetBehaviours (unity3d.com).

Para que empuje al obstáculo hice que guarde a todos los gameObject que hay en la escena con el componente NavMeshObstacle y con el foreach calculo la distancia entre el Thief y el objeto y si está lo suficientemente cerca le aplico una fuerza.

KO

Para entrar en este estado el Guard entra en el animator del Thief y le activa el parámetro de KO. Al entrar en KO, hace el ResetPath()Unity - Scripting API: AI.NavMeshAgent.ResetPath (unity3d.com) para quitar su destino. Para parar el animator directamente hice que se desactive con enabled.

Flee(Worker)

Para hacer el flee del Worker, al inicio busco todos los puntos de agrupación que hay en la escena. Calculo con el Vector3.Distance() y un for la distancia más cercano entre todos los puntos de agrupación. Por consiguiente, hacer que el agente vaya al punto.

Affraid

Para hacer el Worker pueda detectar si dentro de 5 metros hay algún guardia opte a utilizar el Physics.SphereCastAll(), https://docs.unity3d.com/ScriptReference/Physics.SphereCastAll.html. Por consiguiente, con un bucle for y el CompareTag pude cambiar el estado al Posing cuando el guardia está a menos d 5 metros.

Pursue

Para comprobar que el guardia detecta al Thief he tirado un raycast donde lo detecta con el CompareTag. En el caso de true va a la posición del Thief y en otros casos pasa a seek. Además de este raycast he creado otro de 1 metro de distancia para que si detecta al Thief con dicho raycast pasa a Attack.

Attack

De la misma manera que el Pursue, utilice raycast para detectar si el Thief está a 1 metro de distancia. En el caso que sí accede al animator del Thief con el GetComponent y acceder al parámetro del KO.

Para ver si le pierde de vista al Thief he tirado un raycast de 10 metros y si le pierde de vista pasa a seek, si está a más de 2 metros pasa a Pursue.

Al start he hecho que se pare el Guard con el ResetPath() porque quiero que cuando ataque que esté quieto.

Seek

Al entrar en el estado de seek he accedido con el GetBehaviour el m_LastPosition del Guard en el search, que guarda la información del último punto que ha visto al Thief. Luego que haga el SetDestination al m_LastPosition.

Con el raycast si ve al Thief vuelvo otra vez al Pursue, si llega al m_LastPosition y no ve al Thief pasa a Scan

Scan

En el OnStateEnter hice que entre a una corutina desde el script MonoBehaviour del Guard, la cual tiene la función de Rotar 360 grados. Para que rote 360 grados he procedido usar el Math.Lerp https://docs.unity3d.com/ScriptReference/Mathf.Lerp.html. He puesto el tiempo que tarda en completar el lerp igual que el tiempo del while y si a la mitad de ello detecta al

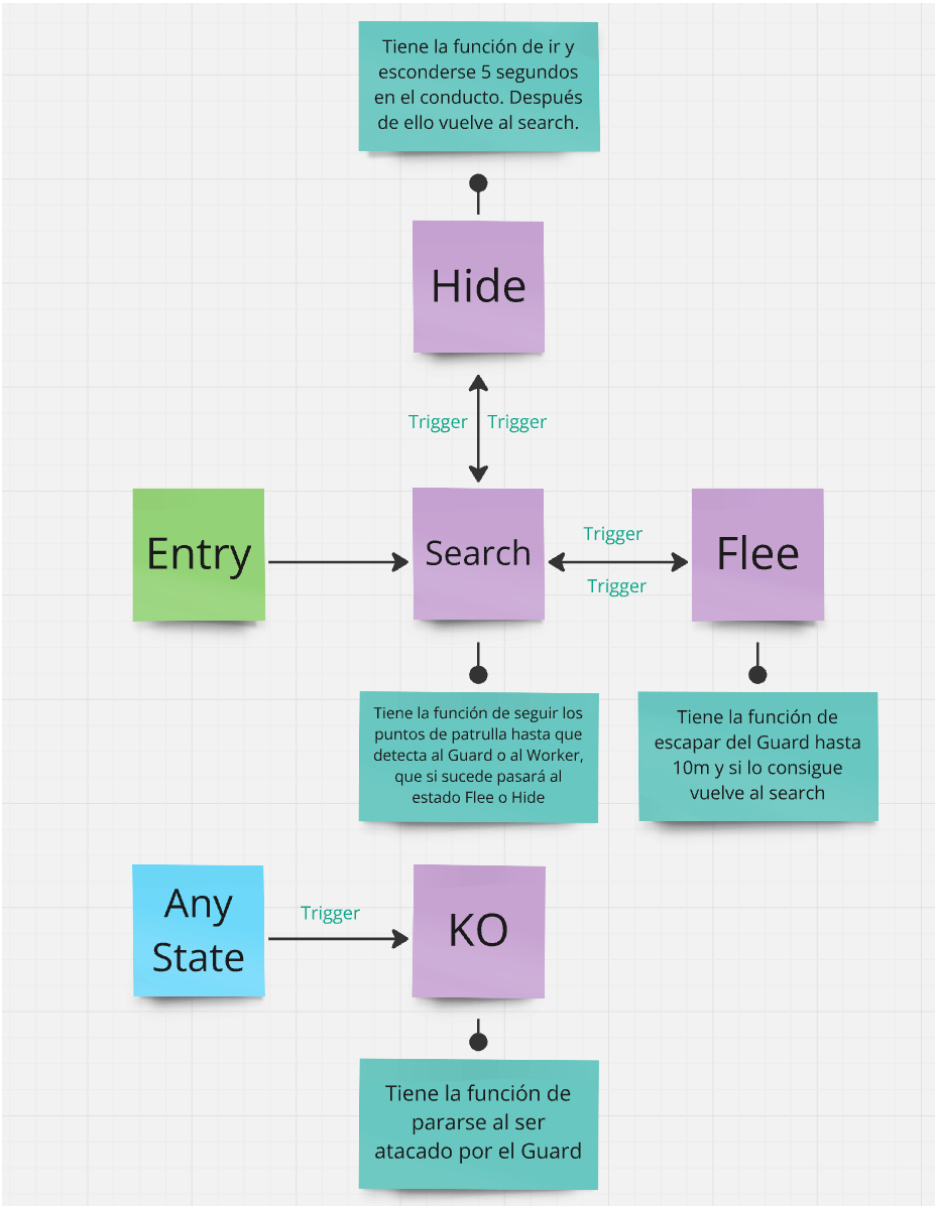
Thief pasa a Pursue saliendo de la corutina con el yield break, en el caso de que no completa la corutina sin detectarlo pasa a Patrol.

Raycast

El raycast que utilice son 2 principalmente, el Physics.Raycast() https://docs.unity3d.com/ScriptReference/Physics.Raycast.html que lo utilizo para similar la visión de los agentes y el Physics.SphereCastAll() en el Worker para ver si hay algún Guard cercahttps://docs.unity3d.com/ScriptReference/Physics.SphereCastAll.html.

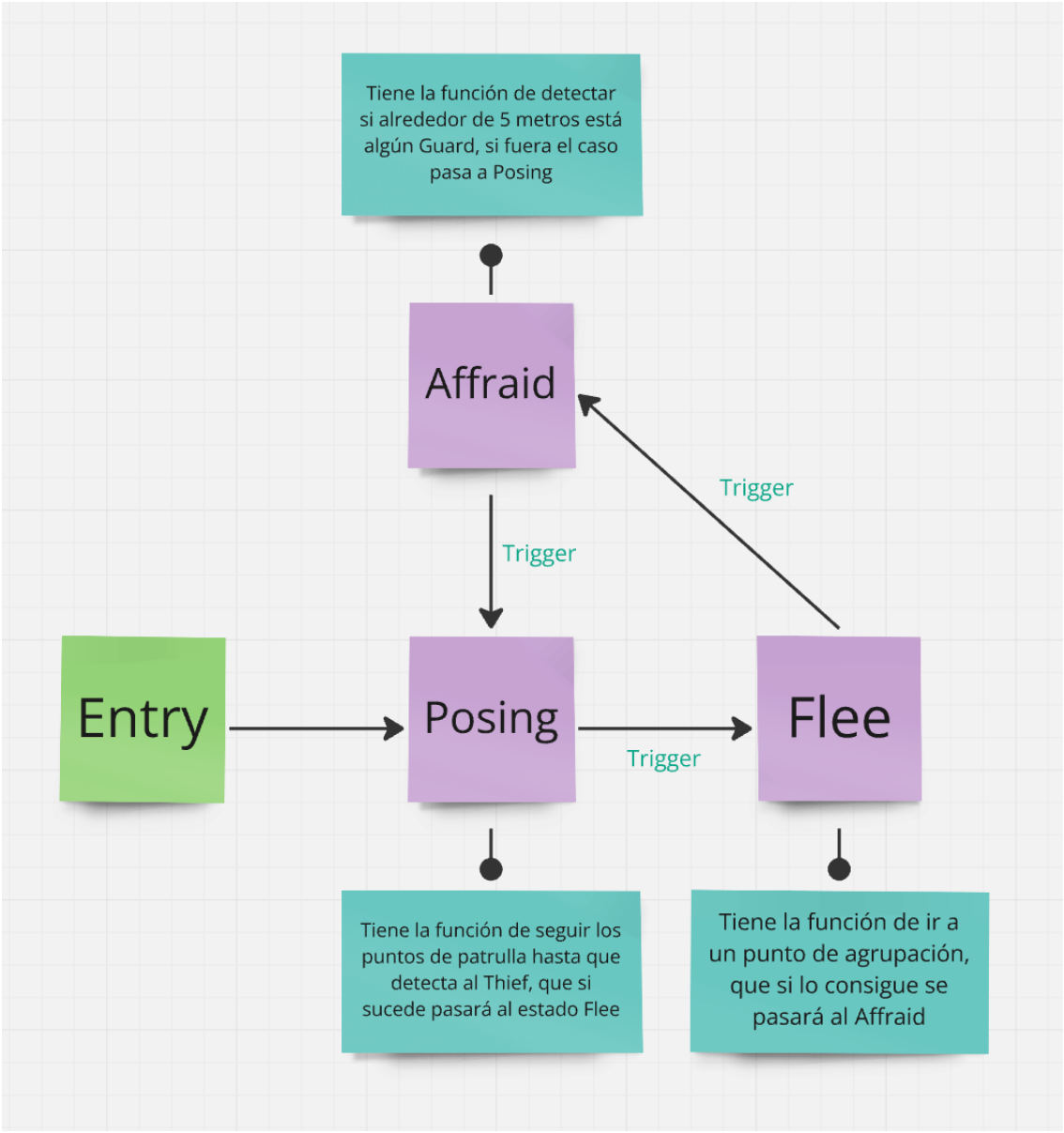
Thief máquina de estado

En todos los parámetros implementé el trigger porque me parecía en este proyecto más cómodos entrar en el estado llamando una única vez.



Worker máquina de estado

En todos los parámetros implementé el trigger porque me parecía en este proyecto más cómodos entrar en el estado llamando una única vez.



Guard máquina de estado

En todos los parámetros implementé el trigger porque me parecía en este proyecto más cómodos entrar en el estado llamando una única vez.