Práctica 1: Navegación

Thief:

Para que el *thief* pueda navegar por todas las área que hay en el escenario utilice el video del *NPCs en Unity movimiento y rutas*, donde he aprendido como en el *NavMeshAgent* puedo asignar el tipo de área donde pueda patrullar, que en el caso de que pueda navegar por cualquier áreas, seleccionó todas.

Guard:

Para que pueda navegar por todas las áreas que existe en la escena, le he asignado al inicio como el mismo parámetro que el thief. Pero posteriormente, hice que no pueda acceder al área duct, ya que solo lo utilizo como área para los componentes *Nav-Mesh Links,* aprendido en la parte del *Navigation Area* del https://docs.unity3d.com/Packages/com.unity.ai.navigation@2.0/manual/NavMeshLink.html. Con la parte del Área tap del apartado https://docs.unity3d.com/Packages/com.unity.ai.navigation@2.0/manual/NavigationWindow.html puedo saber que los links generados automáticamente son del tipo de área jump. Entonces puedo permitir al *guard* acceder al área jump para la caída e impedir el área duct para otros *Nav-Mesh Links*.

Worker:

A diferencia del thief hay que desasignar los áreas a donde puede ir dejando solo el área de walkable. Como sé que los Nav-Mesh Links generadas automáticamente tienen puestos como el área jump y si a los *NavMeshLink* creadas por componentes manualmente les asignó el tipo de área como duct, el *worker* no podrá utilizar ningún *Nav-Mesh Links*.

NavMesh:

Para implementar diferentes áreas he utilizado la información proporcionada en el vídeo del *NPCs en Unity movimiento y rutas*. Así conseguí meter los áreas maintenance y duct, junto con los not walkable y walkable que ya estaban por defecto. También con el vídeo he aprendido a usar el *NavMeshSurface* y *NavMeshModifier*, que en uno hago el bake de la escena(cambiando parámetros como el *Current Object Hierarchy*) y en el otro creo las áreas de tipo maintenance aplicando el *NavMeshModifier* a los gameObject llamado Maintenance que están dentro del gameObject Environment. Utilice la documentación para saber más sobre cómo funciona y cómo modificar en estos links:
https://docs.unity3d.com/Packages/com.unity.ai.navigation@2.0/manual/NavMeshModifier.html
https://docs.unity3d.com/Packages/com.unity.ai.navigation@2.0/manual/NavMeshSurface.html

Patrol:

He realizado la lógica de patrulla en un script llamado *BasicAgent* que se sitúa dentro de la carpeta de Scripts, AIMovement. Utilice el enlace de https://docs.unity3d.com/Packages/com.unity.ai.navigation@2.0/manual/NavMoveToDestination.html para saber como que un gameObject llegue a cierto punto de la escena utilizando el *NavMeshAgent* y https://docs.unity3d.com/Packages/com.unity.ai.navigation@2.0/manual/NavAgentPatrol.html donde

me da un ejemplo de como realizar un script para partullar entre distintos puntos de la escena. Con dichas informaciones he creado en la escena 3 gameObject que guardan los waypoints donde tienen gameObjects con gizmos para asignar diferentes destinos para Guard,Worker y Thief, cada uno de ellos haciendo ver las propiedades de cada uno.

Waypoints del thief: del transform inicial del thief hasta el waypoint 0 se observa como puede atravesar las tuberías. Con el waypoint 1,2 y 3 podemos ver que puede entrar también en las puertas. Posteriormente podemos ver con el waypoint 4 cómo atraviesa la tubería de nuevo. Por último vemos que con el waypoint 5 puede bajar con los links.

Waypoints del worker: del transform inicial del worker hasta el waypoint 0 observamos como el worker coge la ruta más larga ignorando la tubería. del waypoint 0 al 1 vemos que no puede bajar con los links. Del waypooint 2 al 3 podemos ver que tampoco puede atravesar las puertas porque no puede acceder a las áreas del maintenance.

Waypoints del guard: del transform inicial del worker hasta el waypoint 0 observamos que el guard no puede atravesar las tubería. Del waypoint 0 al 1 vemos cómo puede saltar con los links de caída. Con el waypoint 1 al 2 podemos ver que puede navergar por el área del maintenance entrando con la puerta, pero no puede utilizar el link del duct.

Links:

En el caso del link generados automáticamente, como solo funciona el de caída lo que hice fue cambiar en *Navigation Agen*t su *Drop Height* a 5. Esto lo he aprendido en el vídeo del *NPCs en Unity movimiento y rutas* y posteriormente aplicar el bake del *NavMeshSurface*. Por otra parte, para crear los links que funcionan como entrada de tubería apliqué el *NavMeshLink*. aprendido con información del video y el enlace https://docs.unity3d.com/Packages/com.unity.ai.navigation@2.0/manual/NavMeshLink.html.

Obstacles:

He aplicado a todos los prefabs dentro de la carpeta de props el componente *NavMeshObstacle*, que funciona como obstáculo dentro de la escena para que alrededor de ellos sea imposible navegar sin necesidad de hacer el bake. Se le pone el *CarveOnlyStationary* porque son elementos estáticos en la escena. Use el enlace de *https://docs.unity3d.com/Packages/com.unity.ai.navigation@2.0/manual/NavMeshObstacle.html* para saber como son las características de los componentes.

Maintenance Speed:

Para reducir la velocidad tuve que usar el *NavMesh.GetAreaFromName*, para coger la máscara de bit y aplicarlo en *NavMesh.SamplePosition* para comprobar si está dentro de dicha área de navegación(en este caso el maintenance). Esta información la he encontrado en estos 2 links donde me explica como funciona cada uno de ellos y como lo podría combinarlo: https://docs.unity3d.com/ScriptReference/AI.NavMesh.GetAreaFromName.html , https://docs.unity3d.com/ScriptReference/AI.NavMesh.SamplePosition.html. Luego simplemente compruebo que al estar dentro del área maintenance disminuyó la velocidad.