La Guardia

COMPORTAMIENTO DE PERSONAJES

Grupo_Isusti

Pablo Alconchel Palma

Tabla de contenido

Descripción General

La práctica presentada para la asignatura consta de una simulación en la que se ha buscado presentar una gestión automatizada de la guardia que defiende una ciudad. Para ello, se ha creado un modelo de una ciudad y un controlador del movimiento de la cámara para que el espectador se pueda mover libremente con la escena (Funciona igual que la cámara de Unity).

Controles: WASD para moverse, mantener click derecho para rotar y shift para acelerar.

Dentro del propio funcionamiento de la práctica, nos encontramos con varios agentes que ejercen de puntos de interés y de los propios guardias que van a defender la ciudad. Estos mismos, se irán repartiendo por los diversos puntos de defensa de la ciudad (GuardPoints), se quedarán a la espera de que se les asigne un nuevo puesto o se irán a cubrir sus necesidades básicas para poder estar en condiciones de trabajar.

Por otra parte, el resto de agentes que marcan los puntos de interés son muy parecidos a diferencia de los puestos de guardia, que se conectan automáticamente al resto de vecinos que se encuentren dentro de su rango de acción. Con ello se pueden montar grafos automáticos que forman una red de puestos de guardia, donde se priorizan los puestos que más puestos de guardias cercanos tengan, dado que se espera que una zona que requiera más defensa es más prioritaria que una que está sola.

Por último, estarían las zonas de descanso que recrearían las tabernas o cuarteles de los que constarían los soldados para que puedan reponerse. Estos, están esparcidos por varios puntos de la ciudad, por lo que los guardias buscarán el puesto más cercano para saciar sus necesidades antes de volver al trabajo.

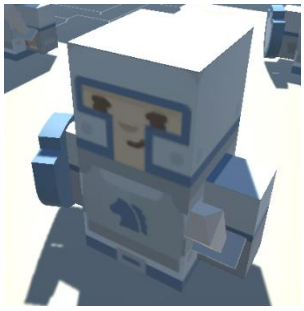
Como resumen de los conceptos aplicados en la práctica nos quedaría la siguiente lista:

- NavMesh: Para que los guardias se puedan mover.
- Objetos Inteligentes: El manejador de GuardPoints ayudan a indicar al guardia que hacer. Mientras que los propios GuardPoints o las zonas de descanso se autogestionan sin necesidad de que el guardia haga algo y están pensados en base como waypoints ordenados en un grafo.
- Percepciones simples: Se usa variables internas dentro de los guardias para representar sus necesidades.

Link al Github: https://github.com/Poxterx/ComportamientoDePesonajes

Descripción de los Agentes

GUARDIA



Nombre: Guardia

Tag: Guard

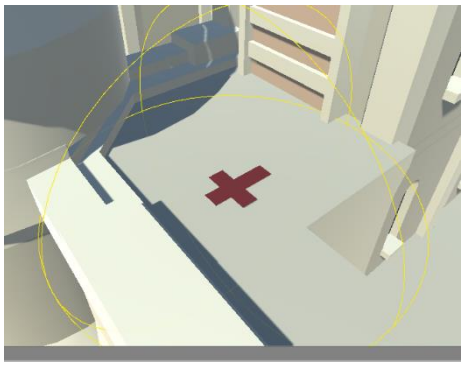
Prefab: Assets/Prefabs/NPCs/Guardia

Descripción: Los guardias son aquellos agentes encargados de defender la ciudad. Por ello deberán buscar ser asignados a un puesto de guardia vacante para poder hacer su trabajo. Si no hubiese trabajo disponible se deberá quedar en reserva a la espera de que quede un puesto disponible. Pero no podemos olvidarnos de que son humanos y tienen necesidades básicas que deben suplir si llegan a estar cansados o hambrientos, por lo que deberán dejar sus puestos para ir a reponerse.

Tabla de Percepciones		
Hambre	Se trata de una variable que va decayendo con el tiempo. Esto mismo ocurre en su método consumo	El único que tiene acceso es el propio guardia, aunque se ha puesto público para verlo en el inspector de UNITY
Energía	Se trata de una variable que va decayendo con el tiempo. Esto mismo ocurre en su método consumo	El único que tiene acceso es el propio guardia, aunque se ha puesto público para verlo en el inspector de UNITY
Tabla de Acciones		
comprobarEstado	Se trata de una comprobación con ifs	Mira si está operativo tras el consumo de sus necesidades
irAPosicionAsignada	Con la Navmesh se mueve a la posición de destino	Dirige al guardia a una nueva posicion
encontrarLugarDeDescanso	Busca con un for la "taberna" más cercana.	Asigna la zona de descanso como nuevo objetivo

FSM

PUNTO DE GUARDIA



Nombre: GuardPoint

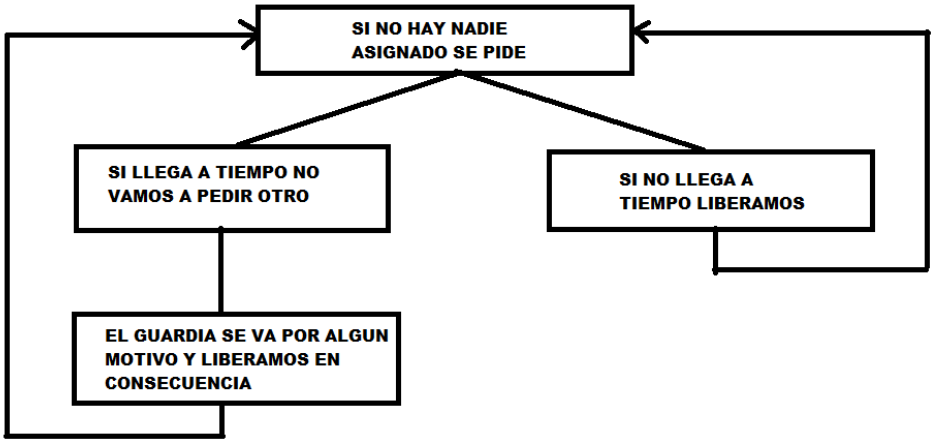
Tag: GuardPoint

Prefab: Assets/Prefabs/Objects/GuardPoint

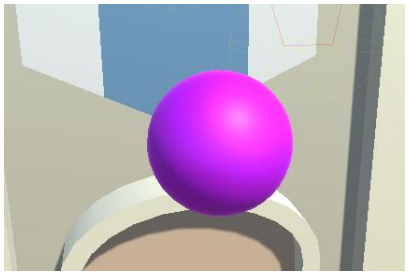
Descripción: Se trata del objeto que se encarga de marcar el lugar exacto donde se debe poner un guardia. El mismo detecta si el guardia que se le ha asignado ha llegado o no en un intervalo de tiempo. También, detecta si el mismo se va para poder pedir otro.

Tabla de Percepciones		
GuardiaAsignado	Utilizamos un collider para saber si el guardia asignado ha llegado/se va.	El propio guardpoint comprueba que el guardia asignado coincide con quien llega al lugar.
Tabla de Acciones		
detectarVecinos y calcularPrioridad	Detectamos al resto de grafos dentro del área para montar el grafo y la prioridad	Los Guard Points se interconectan para montar el grafo en base a sus vecinos.
onTriggerEnter/Exit	Viene de Unity	Nos sirve para detectar la llegada/salida del guardia y actuar en consecuencia.
Update y tiempoDeEspera	Usamos una cuenta atrás para esperar al guardia	Si no llega el guardia a tiempo, se libera y se pide otro.

FSM



MANEJADOR DE PUNTOS DE GUARDIA



Nombre: GuardsManager

Tag: Untagged (es Singleton)

Prefab: Assets/Prefabs/Objects/GuardsManager

Descripción: Es el agente encargado de darle a los guardias una nueva posición que vigilar si hay disponibles o de darles la orden de quedarse en la zona de espera.

Tabla de Percepciones		
Guardias y GuardPoints	Cada vez que un guardia solicita una nueva tarea, se encarga de comprobar si hay trabajo disponible.	Singleton. Todos los guardias pueden acceder a él y sus métodos que les devuelven las nuevas órdenes.
Tabla de Acciones		
montarListaGuardPoints	Guardamos todos los GuardPoints encontrados por su Tag	Usamos esta lista para asignar trabajo y comprobar el estado de cada GuardPoint
mirarTrabajoDisponible	Recorremos la lista hasta encontrar una vacante o no encontrar nada.	Si encuentra trabajo, informa al guardpoint y al guardia para que se conecten
solicitarPuntoDeEspera	Devolvemos la posición del waypoint del punto de espera.	Si un guardia no ha conseguido trabajo, solicita la posición del punto de espera

FSM

Esta clase no contiene un diagrama. Solo maneja los guardPoints.

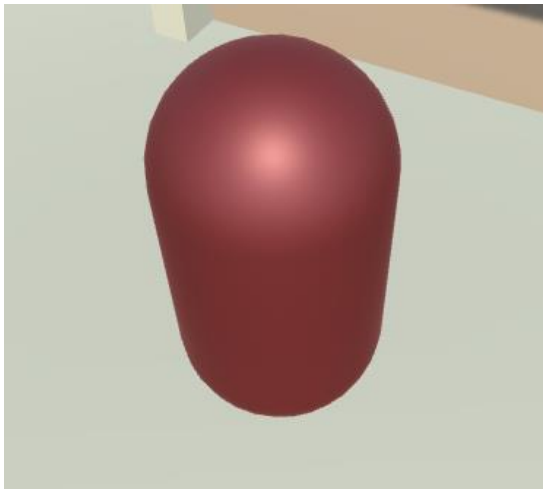
AREA DE DESCANSO Y LUGAR DE ESPERA

Nombre: Reserva

Tag: Reserve

Prefab: Assets/Prefabs/Objects/Reserva

Descripción: Es un objeto exclusivamente creado para hacerr visible la posición del lugar de espera para los guardias. Lo usa el GuardsManager para pasar la posición a los guardias.



Nombre: RestArea

Tag: RestArea

Prefab: Assets/Prefabs/Objects/RestArea

Descripción: Agente encargado de hacer que las necesidades de los guardias, que estén dentro de su rango, incrementen lo máximo posible antes de que ellos mismos se vayan tras saciarlas.

Tabla de Percepciones		
Área de efecto mediante collider esférico.	Cada segundo que pasa, se recuperan las necesidades de los guardias un poco.	El propio objeto se encarga de modificar las necesidades de cada guardia dentro del collider.
Tabla de Acciones		
OnTriggerStay (De Unity)	Se pasa por todos los guardias dentro del área y se les recupera un poco las necesidades básicas	El objeto solo modifica las variables de los guardias. Pero, los guardias pueden salir y entrar cuando quieran

FSM

Flujo y Estructura de Datos

Tabla de relaciones:

	Guardias	GuardPoints	G.Manager	RestArea
Guardias	-	-	Pide que se le asigne una nueva posición de trabajo	Localiza la posición del más cercano para ir a él
GuardPoints	Detecta si es el guardia asignado al entrar en el área	-	-	-
G.Manager	Otorga una nueva posición de trabajo	Comprueba si están operativos o no para su asignación	-	-
RestArea	Si están dentro del área hace que se recuperen poco a poco	-	-	-

Tabla de variables y estructuras:

Clase	Atributo	Descripción
GuardIA (Script de los guardias)	Floats Hambre y energía	Variables que representan las necesidades básicas y se usan para decir si el guardia está operativo o no.
	currentTarget	Representa la posición destino actual
	Bool operativo	Si el hambre y la energía están por debajo de un umbral se pone a false, lo que hace que se busque un lugar de descanso.
	Bool asignado	Si se tiene un GuardPoint asignado se pondrá a true, lo que hará que no se asigne a otro mediante el GuardsManager.
GuardPoint	Int prioridad	Marca la prioridad de asignación que tendrá el nodo dentro del grafo
	Int Peso	Variable manual para darle más importancia a un nodo
	List <GuardPoint> Vecinos	Estructura de datos que conserva las referencias a todos los vecinos a los que se conecta.
	Float RangoVecinos	Establece el radio en el que se tendrán en cuenta al resto de GuardPoints para unirlos
	GuardIA guardiaAsignado	Guarda la referencia del guarda que haya asignado en ese momento para comprobar cuando llega o se va.
	Bool ocupado	Establece si el GuardPoint ya tiene un guardia asignado o no
	Bool llegadaConfirmada	Establece si el Guardia asignado ha llegado al objetivo dentro del tiempo de espera
	Float tiempoEspera	Establece el tiempo que se esperará al guardia antes de volver a pedir que le asignen uno.
GuardsManagerCore	List<GuardPoint> listaGuardpoints;	Estructura de datos de todos los GuardPoints del escenario ordenados por prioridad descendente
RestPoint	Float RecoverRate	Establece la recuperación por segundo de los guardias.

Comportamientos emergentes

La propia implementación de los guardias hace que cada cierto tiempo se produzca una especie de "cambio de guardia" al todos tener unos valores iniciales parecidos de sus necesidades básicas.

Reparto de tareas

Al ser el único integrante del grupo, he tenido que realizar solo todo el trabajo que tiene la práctica. Eso incluye tanto planteamiento de la IA como el modelado de los escenarios y la creación de los prefabs necesarios.

Lecciones aprendidas

En sí, la práctica representa bien lo que tenía en mente de la simulación de la guardia de una ciudad medieval, aunque es cierto que me he ido dando cuenta de muchos aspectos que no había pensado y también muchas situaciones que no tenía previstas. Por otro lado, he visto diversos puntos de la práctica que serían mejorables y más eficientes, dado que muchas de las complejidades programadas son bastante costosas si se hacen a gran escala.

También a la hora de ponerme a programar un poco me he ido dando cuenta de lo poco familiarizado que estoy con unity (dado que soy de un perfil de diseño y no he tocado mucho la programación en Unity). Algo que creo que se ve a la hora de la facilidad que tuve a la hora de montar los escenarios y los prefabs en comparación con la parte de programar y asignar código. Ahora que estamos terminando la carrera y que no tenemos tantas asignaturas encima espero poder ponerme a trastear con este entorno.

Por parte de las clases, me ha parecido bastante interesantes muchos de los planteamientos a la hora de hacer IA, pero también me he dado cuenta de que es bastante más complicado plasmarlo en la práctica.

Licencias

Lo único que he usado externo ha sido:

- Assets de Kenney / FreeToUse
- Código de cámara libre / MIT license
- ProGrids y ProBuilder en Unity.

La licencia del proyecto es GPLv3, es libre bajo las mismas condiciones.