Bags, Please!

MEMORIA DEL PROYECTO

COMPORTAMIENTO DE PERSONAJES

HARD BREAK DEVS

(grupo 5)

Fernando Moreno Díaz

Denis Gudiña Núñez

Luis Miguel Moreno López

ÍNDICE

DESCRIPCIÓN GENERAL

Bags, Please! es un juego de gestión en el que se controla un supermercado.

El objetivo será mantener la satisfacción de los clientes, de manera que podremos dar **órdenes** a nuestros empleados para que **repongan** los artículos.

Se comienza con una cantidad fija de **dinero**, la cual variará en función del rendimiento de nuestra tienda. Es decir, por el balance entre **compras** de clientes y **gastos** de reposición.

Nuestros clientes llevarán una **lista de la compra** e irán visitando los estantes en busca de sus productos. Si se agotara el artículo, no podrán llevárselo, reduciendo su compra y, por tanto, nuestra facturación.

Será tarea del jugador comprar productos para el almacén, de forma que los **reponedores** vuelvan a colocar los artículos. Cuando éstos puedan, irán al **almacén**, cogerán los productos y lo dejarán en su estante correspondiente de forma automática. Comprar nuevos artículos para reponer supondrá un **coste extra** a nuestro dinero.

La lista de cada cliente es desconocida para el jugador, lo cual le deja con la tarea de decidir qué es mejor en cada momento.

Aunque reponedores y clientes sean **independientes**, estarán moviéndose por el escenario simultáneamente, de forma que se **esquivarán** mutuamente si sus trayectorias se cruzan.

Interactividad

Todo el control sobre el juego se establece mediante una **interfaz visual** manejada con el **ratón.**

La acción principal del jugador es **comprar** nuevos alimentos para el **almacén**, al que los reponedores irán continuamente si hay artículos pendientes.

Repositorio

Para el trabajo diario hemos usado un repositorio público en **Github**:

https://github.com/FernandoMoreno98/Bags-Please

DESCRIPCIÓN DE LOS AGENTES

Clientes

Son el grupo más numeroso. Entran al establecimiento en busca de los productos de su **lista de la compra**, única para cada uno y generado de forma aleatoria. Ésta no es visible por el jugador y tiene un límite de cinco artículos.

Una vez entren a la tienda y crucen la pasarela de entrada, buscarán por las estanterías los productos de su lista, uno por uno. Escogerá el **estante más cercano** que contenga alguno de los productos de su lista y no esté agotado. Tras acabar su compra, irá directo a la salida. El importe gastado pasará a nuestro dinero. Evidentemente, cuanto más puedan comprar más obtendremos.

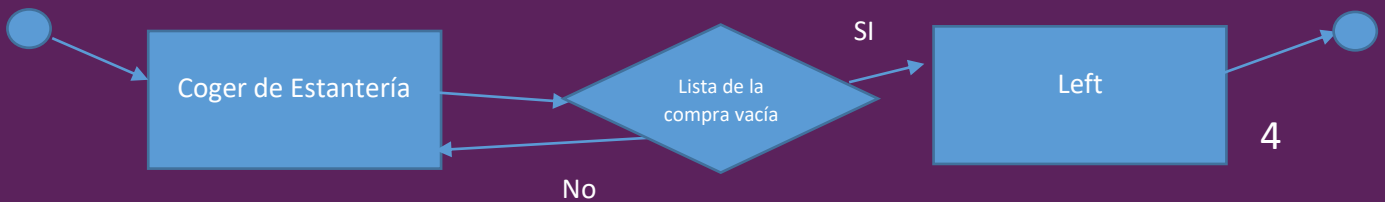
Tabla de percepciones

Nombre	Implementación	Acceso
Visión global de Estantes	Todos los estantes tienen un componente que los identifica	A través de un GameObject Find, y buscamos aquellos objetos con el componente estante.
BackPack	Componente del GameObject	Nos permite saber que comida y cuanta tenemos. Se accede a través del componente
Referencia a la Salida	La salida tiene un waypoint que lo identifica	A través de buscar un componente de tipo salida

Tabla de acciones

Nombre	Implementación	Efectos
Abandonar (Left)	A partir de un Waypoint en la puerta del supermercado. El GOAP determina que esta es la acción más adecuada para conseguir su meta y establece al navmesh agent su destino	El actor se dirige a la entrada del supermercado para posteriormente ser eliminado. Además, dará dinero en función de los productos que lleve.
Coger de Estantería (TakeFoodFromStand)	A partir del componente que tienen todos los estantes para identificarse Se elige el estante más cercano con el alimento que deseamos recoger	El actor se dirige hacia la estantería que cumpla con el requisito y toma la cantidad que necesite de dicho producto (o hasta vaciar existencias)

Diagrama de máquina de estados



Reponedores

Se encuentran siempre dentro del supermercado. Ellos son los encargados de ir al **almacén**, obtener los productos comprados, que se allí se guardan, y llevarlos hasta la **estantería** correspondiente. Pueden coger en cada viaje un número aleatorio de cada tipo de producto que espere en el almacén.

Si el almacén estuviera vacío, esperarán sin hacer nada.

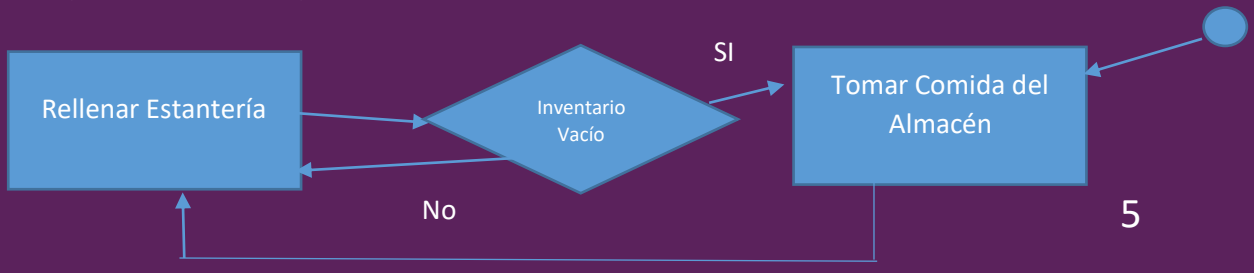
Tabla de percepciones

Nombre	Implementación	Acceso
Visión global de Estantes	Todos los estantes tienen un componente que los identifica	A través de un GameObject Find, y buscamos aquellos objetos con el componente estante.
Referencia al Almacén	El almacén tiene asociado un componente que lo identifica	A través de buscar un componente de tipo WareHouse
BackPack	Componente del GameObject	Nos permite saber que comida y cuanta tenemos. Se accede a través del componente

Tabla de acciones

Nombre	Implementación	Efectos
Rellenar Estantería (FillFoodFromStand)	A partir del componente que tienen todos los estantes para identificarse En primer lugar se intenta rellenar estantes vacíos En segundo lugar, si esto no es posible prioriza estantes medio vacíos con producto que tengamos en la mochila	El actor se dirige hacia la estantería elegida, una vez cerca rellena con la cantidad oportuna (teniendo en cuenta máximos del estante y la cantidad en la mochila) Por tanto pasa la comida de su inventario a la estantería
Tomar comida del WareHouse (TakeFoodFromWareHouse)	A partir del componente se identifica el almacén, el almacén cuenta con su propio inventario, por tanto El reponedor cogerá una cantidad aleatoria limitada de varios productos .	El actor se dirige al almacén, una vez cerca accede a su inventario y por tanto toma productos de este para rellenar el suyo, después vuelve a rellenar estantes.

Diagrama de máquina de estados



FLUJO Y ESTRUCTURAS DE DATOS

El entorno consta principalmente de dos objetos inteligentes que son los estantes y el almacén. Ambos elementos constan de un componente dirigido a hacer de inventario.

En el caso de los estantes, su propio componente identificador guarda tres variables, un enum , dos int que actúan como tipo de alimento que guarda, máxima cantidad permitida , cantidad actual .

Por otro lado el wareHouse hace uso del componente Backpack que también tienen los agentes creados.

Este componente consta de una lista que guarda un tipo alimento (enum) , pero además y para facilitar algunas funciones , genera un diccionario que facilita a partir de un alimento , la cantidad disponible que hay de ese en la lista (ya que al actualizar la lista se actualiza este)

Además, debido a la utilización de GOAP para determinar algunas acciones se debe tener en cuenta el estado del mundo es aquí, en el método de los agentes getWorldData donde se guarda, por ejemplo , si la mochila está vacía o no.

Esto se guarda en un HashSet que enlaza un string como llave a un valor (object).

REPARTO DE TAREAS

Durante el desarrollo hemos desempeñado nuestras tareas de la siguiente manera:

Fernando Moreno

Se encargó del diseño del juego, así como de la creación de *assets* específicos del juego -2D y 3D- y de la búsqueda e integración de los paquetes externos.

También realizó tareas de gestión de equipo, el vídeo explicativo, la memoria, la dirección de sonido y la identidad visual. Autor de la idea original.

Denis Gudiña

Fue programador del sistema de *navmeshes* y de la interfaz. Colaboró en la gestión del equipo y en la memoria, además de implementar el sonido del juego.

Luis Miguel Moreno

Desarrollador de la lógica de los agentes, encargándose de la percepción, de la toma de decisiones y de las relaciones entre ellos.

LECCIONES APRENDIDAS

En este proyecto hemos asimilado una, para nosotros, nueva manera de diseñar un videojuego.

Lo usual fue comenzar a plantear un proyecto por los objetivos o las acciones del jugador. En este caso, comenzamos a diseñar primero un entorno, luego agentes que lo modelaran y, por último, la interactividad del jugador con todo ello. Podríamos decir que, para este caso, la inteligencia artificial pasó de ser un complemento a ser la pieza central del desarrollo.

Posiblemente el punto fuerte de nuestro juego es que emplea una simulación entre agentes para crear una experiencia interactiva y entretenida. Todo ello siendo una escena sencilla, de pocos elementos, pero suficientemente vistosa y llamativa.

Desde una visión crítica, esa sencillez puede llevarlo a ser un juego que aburra pronto. Se podría haber añadido mayor complejidad con más agentes u otorgando a los disponibles una mayor variedad de acciones o estímulos.

En definitiva, ha sido un interesante cambio de perspectiva frente al desarrollo de un juego, siendo el desarrollo de una IA el elemento central que define al resto, logrando convertir una escena sencilla en una experiencia llamativa.

LICENCIAS

Estos han sido los recursos externos empleados en esta práctica.

Algunos de ellos se encuentran en la Asset Store de Unity con licencia gratuita. Otros provienen de páginas externas, pero son de uso libre.

- Artículos del supermercado: ***Food Pack***, Lumo-Art 3D y ***Adorable 3D Food Set***, Layer Lab.
- Base de los personajes: ***Character Pack Sample***, Supercyan.
- Suelo y prototipado: ***Probuilder***, Unity Technologies.
- Música: ***Waltz for a memory***, Mela (FMA).
- Sonidos: ***Freesound*** y ***Free Casual Game SFX Pack***, Dustyroom.

El resto del material no citado es de producción propia.

BIBLIOGRAFÍA

Referencias utilizadas aparte del material de la asignatura.

- GOAP: https://gamedevelopment.tutsplus.com/tutorials/goal-oriented-action-planning-for-a-smarter-ai--cms-20793
- NavMesh: https://github.com/Brackeys/NavMesh-Tutorial