# CDP – MARSHALLOW: PILFERAGE IN YOLKTOWN

**GRUPO 2:** Mario Belén Rivera, Sergio Cruz Serrano, Mireya Funke Prieto, Samuel Ríos Carlos y Enrique Sánchez de Francisco

COMPORTAMIENTO DE PERSONAJES 2020-2021 URJC

# ÍNDICE

# DESCRIPCIÓN GENERAL

## INTRODUCCIÓN

*Marshallow: Pilferage in YolkTown* es un juego 3D de puzles e investigación para un jugador ambientado en un poblado que celebra sus fiestas. En cada partida se deberá encontrar a un ladrón que, aprovechando el jolgorio, deambulará por el pueblo robando a los aldeanos. *Marshallow*, el marshal, tiene el deber de detener a dicho ladrón, para ello hablará con pueblerinos a los que han robado y otros que han sido testigos de un robo. Ambos perfiles de aldeanos te darán los datos necesarios para encontrar al ladrón mediante descripciones sobre su apariencia, sin embargo, algunos no alcanzarán a verlo bien y sus datos serán de dudosa fiabilidad, dificultando la búsqueda del asaltante.

## ESCENA Y LEVEL DESIGN

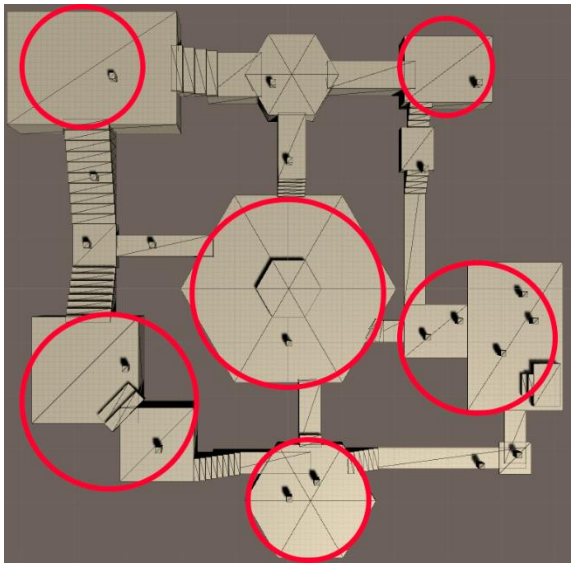
Como se ha comentado antes, el juego transcurre en un pueblo, y este está dividido en seis zonas diferenciadas: el parque, el ayuntamiento, la forja, la plaza, el mercado y las viviendas. Cada zona tiene una forma distinta, algunas son más alargadas y estrechas, y otras como la plaza tienen más espacio, pudiendo albergar a un número mayor de aldeanos. También cabe decir que las hemos repartido en cuatro niveles de altura, estando el parque en la zona más baja y el ayuntamiento en la más alta, añadiendo un componente de verticalidad al juego. Todo lo comentado aporta más variedad no solo visual si no también jugable, ya que cambian tanto las formas de las zonas como los elementos de estas, pudiendo encontrar más maleza en el parque, en contraposición a la plaza donde lo más destacado serán los bancos, las farolas y una fuente central, variando así el movimiento tanto en las zonas como entre ellas. Por tanto, cabe decir que prácticamente toda la información del entorno es estática, ya que todos los elementos que lo conforman son, como se ha dicho, objetos sólidos como casas, árboles, farolas y estructuras de piedra que nunca se van a mover. Ahora bien, también podemos encontrar información dinámica, y esto es debido a que todos los aldeanos cuentan con un cono de visión (que más adelante se explicará en detalle), y dicho cono puede verse afectado si otro aldeano pasa por delante, ya que la visión del primero en ese caso sería parcial y no total.

El diseño de nivel de *Marshallow: Pilferage in YolkTown* es algo que se ha tenido claro desde prácticamente el primer momento, ya que se quería crear un escenario no demasiado grande para concentrar la acción de robo del ladrón y a la vez un lo suficientemente amplio como para dar sensación de libertad al jugador.

Se ha decidido diseñar un nivel en el que las 6 zonas de interés están distribuidas en forma de estrella con una de ellas en el centro (la plaza del pueblo) y las cinco restantes alrededor de esta. Al ser un nivel compacto, se genera una jugabilidad más frenética ya que no hay mucha distancia entre una zona y otra, intentando así no alargar demasiado las partidas.

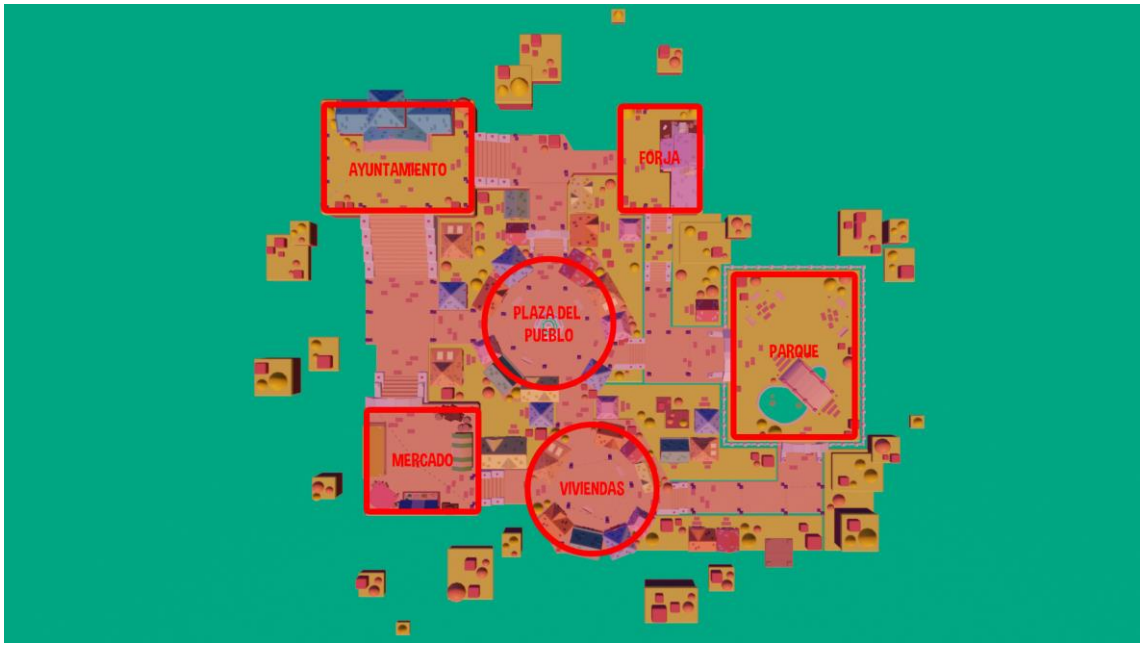
Esta distribución se aprovecha de tal forma que el número de aldeanos por zona sea equitativo, es decir, que no se dé el caso de que haya un exceso de habitantes o falta de ellos en una zona. También se tendrá en cuenta la distribución a la hora de que el ladrón decida robar, ya que por norma general lo hará dentro de dichas zonas (esto se explicará más a fondo en la descripción detallada de los agentes).

A continuación, se muestra una imagen en vista cenital del *blocking* del escenario, resaltando en rojo las zonas comentadas anteriormente. Se puede observar cómo encontramos múltiples conexiones entre estas, generando así un *gameplay* fluido.



Cabe comentar que hemos interconectado todas las zonas desde varios puntos para así aumentar la dificultad a la hora de encontrar al ladrón, debido a que de darse el caso en el que sólo hay un pasillo de salida desde una de las zonas, el ladrón solo puede haber escapado por este y cabe la posibilidad de que se encuentre al jugador, haciendo muy fácil y aburrida la experiencia de juego.

Vista cenital del resultado final del escenario:




## COMPORTAMIENTOS PLANTEADOS

- **Aldeano:**
  - **Aldeano que ha sido robado (víctima):** este PNJ te dará siempre dos datos sobre el ladrón.
  - **Aldeano que presencia un robo (testigo):** este PNJ sólo te da un dato sobre el ladrón.
  - **Aldeano viandante:** el aldeano se mueve de un punto clave del poblado (forja, ayuntamiento...) a otro
  - **Aldeano merodeando:** merodea por una zona.

    - **Aldeano arrestado:** si el marshal le selecciona se para y reacciona a la decisión de este.
- **Ladrón:** roba a aldeanos, dejando un margen de tiempo entre robo y robo para no levantar sospechas. Además, puede despistar al marshal dando datos falsos haciéndose pasar por un testigo.
- **Marshal:** personaje jugable que se mueve libremente por el escenario pudiendo obtener datos del ladrón. No se trata de un agente inteligente, pero a la hora de moverse utiliza una *Navmesh* que determina el camino a seguir hasta el punto indicado y si se puede o no alcanzar.

### CONTROLES

Para mover al personaje por el escenario, encontramos dos posibilidades, dependiendo de si se juega a *"Marshallow: Pilferage in YolkTown"* en ordenador o en dispositivo móvil:

- El control en ordenador es mediante *clicks* de ratón, haciendo *click* en la parte del escenario a la que se desee moverse. Además, este mismo control es usado para todo, ya sea navegar por la interfaz en los menús o para girar la cámara en mitad de partida.
- En cuanto al control en dispositivos móviles, es exactamente igual que en ordenador, pero sustituyendo los *clicks* por "*taps*" o toques en la pantalla.

### DIFICULTAD Y TESTEO

Finalmente, cabe destacar que hemos ideado tres niveles de dificultad (fácil, media y difícil). Esto repercute en la jugabilidad dado que muchos de los parámetros del juego cambian (como el número de aldeanos, las probabilidades de que los datos que dan los aldeanos sean ciertos o dudosos…). Se ha creado un apartado extra en este documento donde figuran dichos parámetros y también los cambios que han sufrido durante la fase de testeo para que el resultado final sea de una Inteligencia artificial medianamente compleja, pero sobre todo divertida.

### REPOSITORIO DE GITHUB

https://github.com/lightningopal/Juegos-para-Web-y-Redes-Sociales---Unity

## DESCRIPCIÓN DE LOS AGENTES

*Marshallow: Pilferage en Yolktown* cuenta con dos tipos de PNJs, los aldeanos y el ladrón. A modo de introducción, explicaremos resumidamente el funcionamiento del videojuego y qué papel tienen dichos PNJs. Todos los personajes cuentan con accesorios, colores del cuerpo y número de ojos distintos al resto, por tanto, cada aldeano tiene sus características distintivas. La gracia es que el ladrón contará con sus propios accesorios, y al ser visualmente igual que un aldeano, se mimetizará entre ellos. Y, por si fuera poco, cada partida es diferente al resto, ya que todos los aldeanos aparecerán con características aleatorias, teniendo en cuenta que no puede haber dos iguales. Para identificar al ladrón, que no es tarea fácil, el usuario hará uso de las declaraciones de los habitantes que han sido tanto víctimas como testigos de un robo, y una vez recabada cierta información el jugador será capaz de diferenciar al ladrón entre todos los aldeanos para así detenerlo y ganar la partida.

Para el comportamiento de los aldeanos y el ladrón, hemos decidido utilizar *behaviour trees*, ya que son una solución de alto nivel fácil de trabajar y escalables, que permiten cambios de manera rápida y cómoda. Además, son bastante eficientes ya que solo comprueban nodos que aún sean posibles.

A continuación, se explicará en profundidad el comportamiento de todos los personajes que conforman el videojuego.

## ALDEANO

En primer lugar, encontramos a los aldeanos, PNJs que transitan por el escenario recorriendo el mapa de un punto de interés a otro, siendo estos las diferentes partes del pueblo, como por ejemplo el parque, el mercado o el ayuntamiento. Por lo tanto, el objetivo de estos agentes será simplemente merodear el pueblo socializando, ya que son las celebraciones de las fiestas. El aldeano elegirá aleatoriamente a qué punto va a moverse una vez lleve 10 segundos en el actual, decidiendo una vez en el nuevo punto si se queda o se mueve a otro. Esto sucede para que no se supere el cupo máximo de posibles aldeanos en un mismo punto, consiguiendo de esta forma un mejor reparto de las zonas, ya que nunca se darán los casos donde haya una zona con exceso de habitantes o con ausencia de ellos. Para moverse entre zonas, el agente utilizará una *Navmesh*.

Como se ha comentado anteriormente, los aldeanos pueden ser tanto víctimas como testigos de robo por parte del ladrón, lo cual quiere decir que su objetivo principal se vería interrumpido. Por tanto, en total contaríamos con cinco tipos de comportamientos de aldeano: arresto, víctima, testigo, merodear y viandante. El estado por defecto de un aldeano es el de viandante o merodear, pero este cambia al de víctima si le roban, al de testigo si presencia un robo o al de arresto si el jugador pulsa sobre él.

Seguidamente se explicarán más a fondo estos posibles estados y cómo se comportan los agentes.

### FLUJO EN PARTIDA DEL COMPORTAMIENTO DE UN ALDEANO (DESDE EL INICIO DE ESTA)

- El habitante aparece de forma aleatoria en el mapa (teniendo en cuenta las restricciones de cupo máximo para repartir equitativamente a los agentes en caso de aparecer en una zona).
- Tras pasar 10 segundos en una zona (si ha aparecido en una) en el estado de merodear, elige la zona a visitar a continuación.
- Transita hacia la zona elegida. El agente decide si hacerlo andando o corriendo. No hay problema si durante el trayecto circula por una zona ocupada por el número máximo de aldeanos.
- Al llegar a la zona comprueba si esta cumple el cupo máximo de aldeanos.
    - Si es así, el agente en cuestión elige de forma aleatoria otra zona de interés del escenario, descartando la zona en la que se encuentra actualmente.
    - Si el número de aldeanos es menor al máximo permitido, el agente entrará en la zona.
- Una vez se encuentra en una zona disponible, pasará al estado de merodear, en el cual interactúa con los elementos y los aldeanos de la zona (la interacción se representa mediante gráficos e iconos que representas emociones elegidas de forma aleatoria).
- Vuelta a empezar desde el paso dos.

Este flujo (perteneciente al estado de viandante y merodear) se puede ver interrumpido por presenciar, ser víctima de un robo o ser arrestado, y su resolución sería la siguiente:

### ALDEANO EN ESTADO DE VÍCTIMA

- Cuando el ladrón fija a un aldeano como objetivo, se acerca hasta él para robarle.
- Una vez ocurra el robo, el aldeano permanece quieto indicando mediante un *bark* (en forma de exclamación roja) que ha sido víctima de un robo.
- Cuando el jugador se acerca a la víctima (es decir, se encuentra dentro del radio de interacción del agente), esta le proporciona información referente al ladrón mediante otro *bark* en forma de

bocadillo con dos iconos (pistas) dentro. Llegados a este punto, el agente decide si facilitar un tipo de información u otro:

- Un 30%* de las veces el aldeano recordará sin problemas las dos características del ladrón que proporcionará al jugador, esto se traduce en que la información será fiable y por tanto las dos pistas serán 100% ciertas. Por ejemplo, si en el bocadillo aparecen una mancha de color azul y una corbata, esto querrá decir que el ladrón es de color azul y porta una corbata con un 100% de probabilidad.
- El otro 70%* de las ocasiones, la víctima no será capaz de recordar nítidamente las características del delincuente. Esto se representa visualmente con un simple icono de interrogación encima del bocadillo informativo. En estos casos también se darán dos pistas, pero no estará asegurado el 100% de la fiabilidad en ambas. Esto quiere decir que una de las pistas será veraz y la otra dudosa, lo cual significa que existirá una probabilidad del 40%* de que esta última sea falsa (cabe decir que el jugador no tiene forma de saber cuál es cierta y cuál no). Por ejemplo, tomando la misma situación del anterior apartado, en este caso habría una posibilidad del 40%* de que o el ladrón no fuera realmente azul o no llevara corbata.

- Cuando el usuario se aleja del aldeano (sale del radio de interacción) este vuelve a su comportamiento normal de viandante o merodear.

*Porcentaje perteneciente al nivel de dificultad media.

A continuación, se muestra el proceso de interacción del jugador con la víctima:


### ALDEANO EN ESTADO DE TESTIGO

Todos los habitantes cuentan con un cono de visión que les permite ver un robo si se diera el caso. Por tanto, suponiendo que alguno ve un delito:

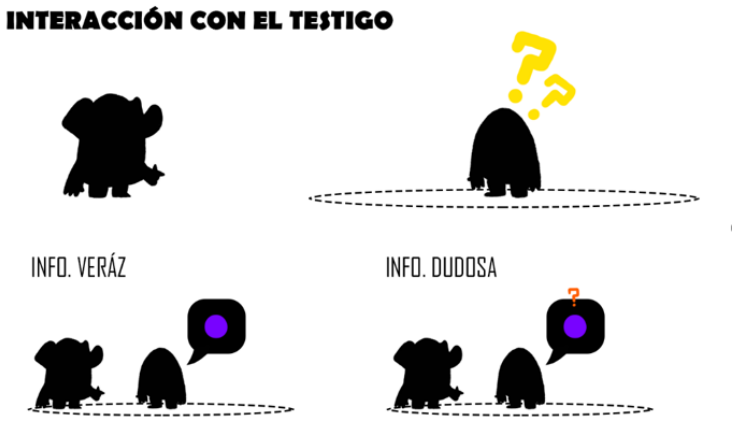
- El agente continúa realizando su comportamiento de viandante o merodear, pero indica con un *bark* en forma de interrogación que ha presenciado el hurto.
- Al acercarse el jugador al testigo (es decir, se encuentra dentro del radio de interacción del agente), este se queda quieto y le proporciona información de la misma forma que la víctima, pero con un pequeño cambio. El aldeano decide si:
  - Facilitar información 100% fiable (un 35%* de las veces). El testigo habrá visto nítidamente UNA característica del ladrón. Por ejemplo, si en el bocadillo aparecen unos cuernos de cabra esto querrá decir que sin lugar a dudas el delincuente tiene cuernos de cabra.

- Proporcionar información dudosa (el otro 65%* de las ocasiones). Esto se representa visualmente con un simple icono de interrogación encima del bocadillo informativo. Aunque ha presenciado el robo, el testigo no habrá podido discernir bien ninguno de los atributos del ladrón, lo cual quiere decir que la única pista será cierta sólo el 50%* de las veces. De nuevo, tomando el mismo ejemplo de antes, en este caso la probabilidad de que el ladrón tenga cuernos de cabra es del 50%*.

- Cuando el usuario se aleja del aldeano (sale del radio de interacción), este vuelve a su comportamiento normal de viandante o merodear. En caso de que un aldeano en estado de Testigo presencie un nuevo robo, perderá su información del robo anterior.

*Porcentaje perteneciente al nivel de dificultad media.

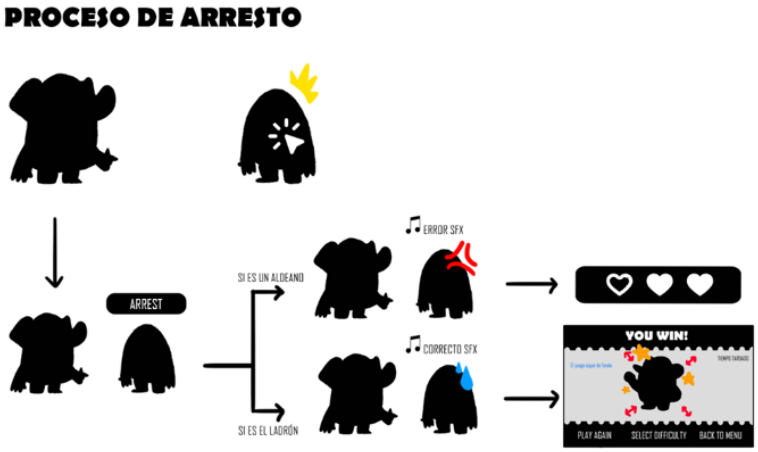
A continuación, se muestra el proceso de interacción del jugador con el testigo:




## ALDEANO EN ESTADO DE ARRESTO

Cuando el jugador selecciona un aldeano cualquiera (hace clic/*tap* sobre él), el aldeano se queda quieto en el sitio y mira a *Marshallow*. Además, aparecerá un botón sobre la cabeza del aldeano que le dará la opción al jugador de arrestarle si este quisiera. Si se selecciona otro aldeano, o pulsa en un sitio para moverse, el aldeano seleccionado en primera instancia continúa su camino y el botón desaparece.

A continuación, se muestra el proceso de interacción del jugador con el aldeano para arrestar:



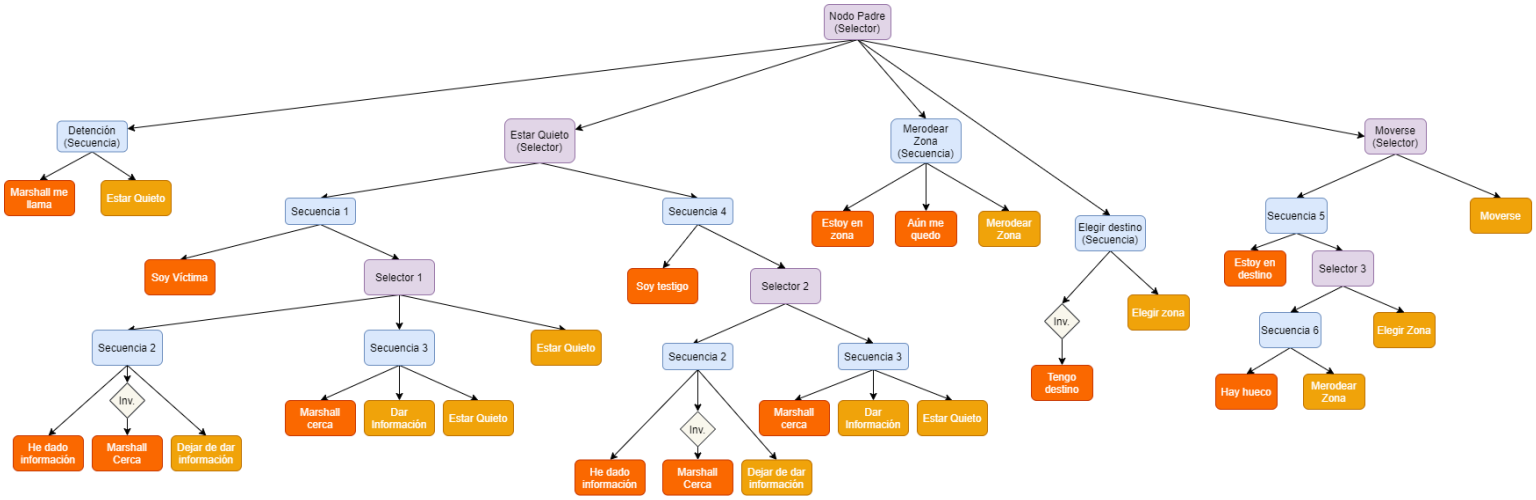
## TABLA DE PERCEPCIONES

| Nombre | Implementación |
|---|---|
| **Percepción visual** | Cono de visión |
| **Percepción de área** | Área de interacción circular con centro en el aldeano |

## TABLA DE ACCIONES

| Nombre | Implementación | Efectos |
|---|---|---|
| **Estar quieto** | Si el agente tiene un camino, lo reseteamos. | El aldeano permanece estático en un sitio concreto. |
| **Merodear zona** | Si el PNJ se está moviendo, pero ha llegado al punto deseado, lo detenemos y lanzamos una probabilidad de 0.3 (30%) de mostrar un emoticono con el estado de ánimo del PNJ.<br><br>Si no se está moviendo, si tiene camino lo reseteamos. Si además ha pasado el suficiente tiempo como para moverse de nuevo, escondemos el emoticono si lo tenía activo y establecemos una nueva dirección para el PNJ. Además, dependiendo de si anda o corre, se establece una u otra animación. | El aldeano se mueve de un lado a otro dentro de una misma zona. A veces pueden aparecer emoticonos que indiquen interacción. |
| **Elegir zona** | Se elige una zona aleatoria que no sea la zona en la que ya estaba y se establece el destino del aldeano hacia allí. Además, se lanza la probabilidad de si va andando o corriendo a su destino. | El aldeano selecciona la siguiente zona a la que se va a mover. Además, elige si ir a dicha zona corriendo o andando. |
| **Moverse** | Si el agente no tiene camino, establecemos su destino al que tenía guardado (esto es así para cuando el marshal lo detiene para ver si es el ladrón, poder retomar su camino). | El aldeano transita hacia la zona elegida. |
| **Dar información** | Se llama al método *ShowInformation* del PNJ y se rota hacia el marshal. | El aldeano proporciona información veraz o dudosa sobre el ladrón. |
| **Dejar de dar información** | Se esconde la información mediante el método *HideInformation* de PNJ. Además, retoma su camino. | El *bark* de información que proporciona el aldeano desaparece y este vuelve a su estado de viandante o merodear. |

## BEHAVIOUR TREE (DIAGRAMA UML)*




## CONDICIONES

- **Marshal me llama:** Si el jugador hace clic sobre un aldeano, se considera que el marshal le está llamando.
- **Soy víctima:** Si el aldeano ha sido robado.
- **Marshal cerca:** Si el jugador entra en el radio de interacción del aldeano.
- **Soy testigo**: Si el aldeano ha presenciado un robo.
- **Estoy en zona**: Si el aldeano se encuentra en una zona de interés, visitándola, no de paso.
- **Tengo destino:** Si el aldeano sabe a qué zona moverse.
- **Estoy en destino:** Si el aldeano ha llegado a la zona que tenía previamente elegida.
- **Hay hueco:** Si la zona en la que el aldeano se encuentra tiene hueco suficiente para entrar.
- **He dado información:** Si el aldeano ya ha proporcionado las características del ladrón al marshal.
- **Aún me quedo:** Si el aldeano todavía no tiene que irse de la zona.

## ACCIONES (EXPLICADAS PREVIAMENTE EN LA TABLA DE ACCIONES)

- **Dar información:** Un *bark* con forma de bocadillo que contiene información sobre el ladrón ya sea información verdadera o dudosa.
- **Estar quieto:** El aldeano se queda en una posición fija.
- **Merodear zona:** El aldeano se mueve libremente por la zona con cierta probabilidad de mostrar mediante emoticonos su estado de ánimo.
- **Elegir zona:** Se elige una zona a la cual se desea moverse, sin poder repetirse la última elegida. Además, se elige si se dirige a la zona corriendo o andando.
- **Moverse:** Transitar hacia la zona elegida.
- **Dejar de dar información:** El aldeano vuelve a su estado de viandante o merodear, desapareciendo así el *bark* de información.

## NODOS DECORADORES

- **Inv.:** Se invierte la condición que se esté comprobando.

*Para un mejor visionado del diagrama, se ha adjuntado una imagen en el entregable.

## LADRÓN

En segundo lugar, encontramos al ladrón, un PNJ que rondará la ciudad con un comportamiento similar al de los aldeanos para así no llamar la atención, sin embargo, el objetivo de este agente es robar a los habitantes del pueblo. Para ello el ladrón deberá encontrarse en una zona de interés y además haber alguna víctima a la que robar dentro de su área de visión, todo esto sin dejar de lado que, si el marshal se acerca demasiado, el robo se cancela y disimula actuando como un aldeano normal. Para moverse entre zonas, el agente utilizará una *Navmesh*.

El ladrón cuenta con dos estados diferentes (y comparte los estados de viandante, merodear y arresto del aldeano), el estado de robo o el estado de fingir ser testigo, siendo el primero donde se da la situación en la que el ladrón actúa como tal y el segundo un estado donde se comporta como un aldeano en estado de testigo.

### FLUJO EN PARTIDA DEL COMPORTAMIENTO DEL LADRÓN (DESDE EL INICIO DE ESTA)

- El ladrón aparece de forma aleatoria en el mapa (teniendo en cuenta las restricciones de cupo máximo para repartir equitativamente a los agentes en caso de aparecer en una zona).
- Tras esperar un breve periodo de tiempo, ejecuta un robo (con las condiciones pertinentes, explicadas más adelante).
- Huye de la zona y decide si disimular siendo testigo de robo o no.
- Vuelta al paso dos.

### LADRÓN EN ESTADO DE ROBO

- En primer lugar, el estado de robo se activa cuando ha pasado cierto tiempo desde el robo anterior para así no llamar tanto la atención y simular un comportamiento realista. Al principio de la partida también hay un pequeño tiempo de espera para robar.
- En segundo lugar, una vez el modo robo se activa, el ladrón comprueba si se encuentra en un punto de interés. En caso de no ser así, busca dicha zona para así ejecutar su robo. En caso contrario, el ladrón elige una víctima.
- Para elegir víctima primero debe de haber aldeanos dentro de su área de visión. En caso de no encontrar a ningún aldeano, éste se quedará en estado de merodear (al igual que el resto de los aldeanos) hasta que se acaben los 10 segundos de merodear o hasta que un aldeano entre dentro de su área de visión. Sin embargo, si hay alguien a quien robar, el ladrón escoge de entre todos ellos a uno aleatorio.
- Cuando la víctima esté seleccionada, el ladrón se dirige rápidamente a robarle (a una velocidad mayor que la velocidad máxima del aldeano, para que así el robo se ejecute de la forma más rápida posible).
- Una vez ambos colisionen, sucede el robo y el ladrón decide si actuar como testigo o no (estado explicado más adelante).
- Si el ladrón decide no actuar como testigo, simplemente ejerce la función de aldeano en estado de viandante de nuevo.

Si durante el proceso de robo el marshal entra dentro de la zona de visión del ladrón, el robo se cancelará y continuará con la acción que estuviese realizando (merodear o viandante), haciéndose pasar por un aldeano normal y corriente.

### LADRÓN EN ESTADO DE FINGIR SER TESTIGO

Una vez ha llevado a cabo un robo, decidirá si pasar a actuar como un testigo mientras deambula de una zona a otra. En caso de que actúe como testigo, un *bark* en forma de interrogación aparecerá sobre su cabeza dando a entender al marshal que ha presenciado un robo (tal y como lo haría el aldeano en estado de testigo). Si el marshal se acerca a él, sucederá lo mismo que sucede cuando el marshal se acerca a un aldeano testigo y este le muestra un *bark* de información dudosa (es decir, el que tiene un interrogante en el bocadillo), la diferencia es que, en el caso del ladrón, esta información tendrá un 100% de probabilidad de ser falsa. La gracia reside en que el jugador no tiene forma de saber esto y simplemente lo confundirá con un testigo normal y corriente.

### TABLA DE PERCEPCIONES

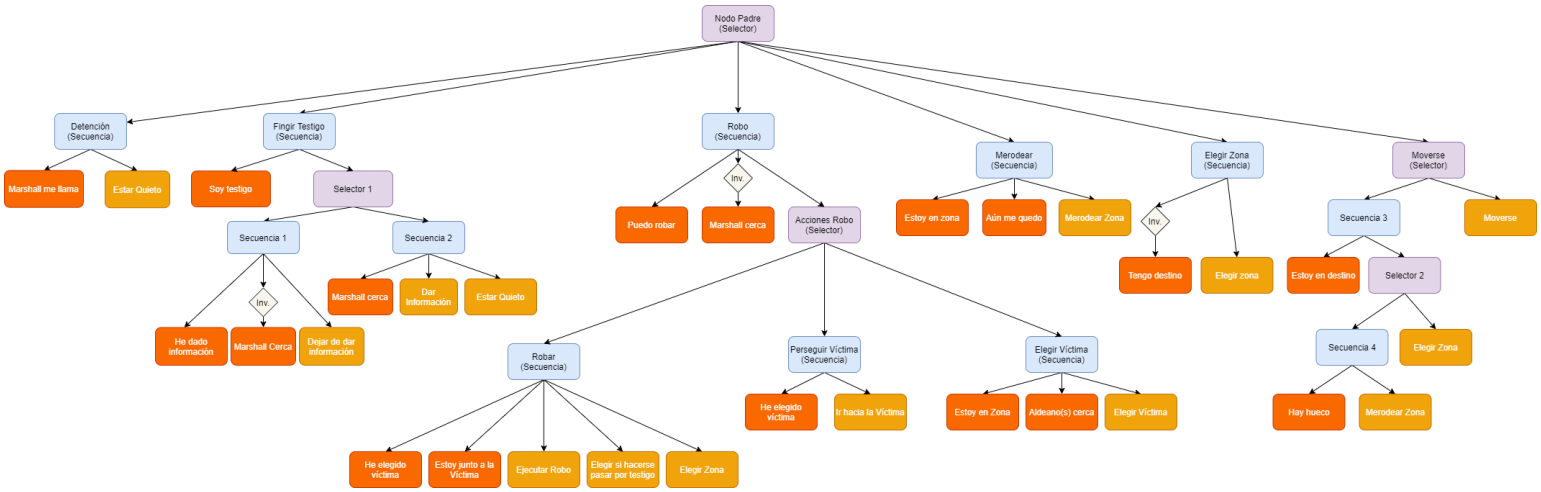
| Nombre | Implementación |
|---|---|
| Percepción de área de marshal (para cancelar un robo) | Área de interacción circular con centro en el ladrón |
| Percepción de área de marshal (para mostrar información en caso de fingir ser testigo) | Área de interacción circular con centro en el ladrón |
| Percepción de área de aldeanos (para elegir una víctima) | Área de interacción circular con centro en el ladrón |

### TABLA DE ACCIONES

| Nombre | Implementación | Efectos |
|---|---|---|
| Elegir zona | Funciona igual que el nodo Elegir Zona del aldeano | El ladrón elige una zona a la que transitar. Además, elige si ir corriendo o andando. |
| Moverse | Utiliza el mismo nodo que el aldeano, *MoveToDestinationNode*. | El ladrón camina hasta la zona elegida. |
| Merodear zona | Utiliza el mismo nodo que el aldeano, *WanderNode*. | Una vez se encuentra en la zona, se comporta como el aldeano, moviéndose de un lado a otro y mostrando emoticonos. |
| Elegir víctima | Escoge aleatoriamente como víctima uno de los aldeanos que están a su alrededor y establece su velocidad a la *STEALING SPEED*. Además, guardamos la zona actual como zona previa y borramos de la información su zona actual y de su destino. | El ladrón escoge a un aldeano al que va a robar. |
| Ir hacia la víctima | Se establece el destino del agente a la posición de la víctima. | Después de haber elegido a la víctima, el ladrón procede a acercarse para efectuar el robo. |
| Ejecutar robo | Se roba al aldeano, se hace un llamamiento al resto de aldeanos para que comprueben si han visto el robo, se | El ladrón roba al aldeano elegido. |

| | | |
|---|---|---|
| | establece el tiempo para el siguiente robo, se calcula el robo y se instancia el icono en la interfaz. Para acabar, eliminamos la información de la víctima de los datos del ladrón. | |
| Elegir si hacerse pasar por testigo | Se lanza la probabilidad de si se hace pasar o no por testigo tras realizar un robo.<br><br>En caso positivo, lo establecemos como testigo y calculamos el objeto que dará erróneamente al jugador en caso de encontrarse con él.<br><br>En caso negativo, lo quitamos de testigo y borramos su *bark* de testigo (esto es necesario por si el ladrón decide hacerse pasar por testigo en el robo 1 y no en el 2, por ejemplo). | Tras efectuar el robo, el ladrón decide si simular el estado de testigo de un aldeano. |

## BEHAVIOUR TREE (DIAGRAMA UML)*




## CONDICIONES

- **Marshal me llama:** Si el jugador hace clic sobre un aldeano, se considera que el marshal le está llamando.
- **Puedo robar:** Si ha pasado suficiente tiempo como para cometer un robo.
- **Marshal Cerca (detección):** Si el jugador se encuentra dentro del área de detección del ladrón.
- **Marshal Cerca (información):** Si el jugador se encuentra dentro del área de dar información del ladrón.
- **He elegido víctima:** Si el ladrón ya tiene objetivo de robo.
- **Estoy junto a la víctima:** Si el ladrón colisiona con la víctima.
- **Estoy en zona:** Si el ladrón se encuentra en una zona de interés, visitándola, no de paso.
- **Aldeano(s) cerca:** Si en el área de detección de aldeanos hay algún aldeano.
- **Tengo destino:** Si el ladrón sabe a qué zona moverse.

- **Estoy en destino:** Si el ladrón ha llegado a la zona que tenía previamente elegida.
- **Hay hueco:** Si la zona en la que el ladrón se encuentra tiene hueco suficiente para entrar.
- **Aún me quedo:** Si el ladrón todavía no tiene que irse de la zona.

#### ACCIONES (EXPLICADAS PREVIAMENTE EN LA TABLA DE ACCIONES)

- **Ejecutar robo:** Robar al aldeano y cambiar el estado de robo a false.
- **Elegir si hacerse pasar por testigo**: El ladrón elige si hacerse pasar por testigo tras un robo.
- **Ir hacia la víctima:** Moverse desde la posición actual hasta la víctima elegida.
- **Elegir víctima:** Seleccionar a qué aldeano robar.
- **Merodear zona:** El aldeano se mueve libremente por la zona con cierta probabilidad de mostrar mediante emoticonos su estado de ánimo.
- **Elegir zona:** Se elige una zona a la cual se desea moverse, sin poder repetir la última elegida. Además, se elige si se dirige a la zona corriendo o andando.
- **Moverse:** Transitar hacia la zona elegida.

#### NODOS DECORADORES

- **Inv.:** Se invierte le condición que se esté comprobando.

*Para un mejor visionado del diagrama, se ha adjuntado una imagen en el entregable.

Como se ha podido ver, todos los agentes sienten, piensan y actúan. Por parte de los aldeanos, extraen información sobre las zonas y la procesan a la hora de razonar si deciden quedarse en la zona elegida o desplazarse a otra, convirtiendo esta información en un acto de decisión. Algo similar ocurre con el ladrón, que debe elegir a qué aldeano robar y cuándo hacerlo, teniendo en cuenta ciertas variables a la hora de actuar.

## FLUJO DE ESTRUCTURA DE DATOS

### TABLA DE FLUJO DE INFORMACIÓN

#### ALDEANOS

| Desde el entorno | Hacia el entorno |
|---|---|
| Posición del robo a la vista | Testigo e Interrogación |
| Víctima o testigo y posición del marshal cerca | Información sobre el ladrón, mirar al marshal y detener movimiento |
| Evento robo | Víctima, exclamación y detener movimiento |
| Evento llamada del marshal | Mirar al marshal y detener movimiento |
| Botón arresto pulsado | Reacción negativa y disminución de vidas |
| Nº aldeanos en zona > max | Cambiar a otra zona |
| Nº aldeanos en zona < max | Entrar y merodear en zona |
| Tiempo en zona > 10 segundos | Cambiar a otra zona |
| Evento cambiar a otra zona | Elegir correr o andar |

### LADRÓN

| Desde el entorno | Hacia el entorno |
|---|---|
| Tiempo desde el último robo, en zona y aldeano cerca | Elegir víctima y puedo robar |
| Evento puedo robar | Correr hacia la víctima y robar |
| Evento robo | Elegir ser testigo o no y cambiar a otra zona |
| Testigo y posición del marshal cerca | Información falsa, mirar al marshal y detener movimiento |
| Posición marshal cerca | Cancelar robo |
| Evento llamada del marshal | Mirar al marshal y detener movimiento |
| Botón arresto pulsado | Reacción nerviosa y fin de partida |
| Nº aldeanos en zona > max | Cambiar a otra zona |
| Nº aldeanos en zona < max | Entrar y merodear en zona |
| Tiempo en zona > 10 segundos | Cambiar a otra zona |
| Evento cambiar a otra zona | Elegir correr o andar |

## ESTRUCTURAS DE DATOS

Para el agente controlado por el jugador (el marshal), contamos con atributos que determinan, por ejemplo, la distancia a la que *Marshallow* se queda quieto frente al aldeano que se ha elegido para arrestar, o parámetros relacionados con la *Navmesh* que afectan al movimiento de este. *****

Respecto a los aldeanos y al ladrón, cuentan con un "Objeto de Información", el cual almacena en dos objetos "*Item*" el nombre, *GameObject* y *Sprite* del objeto a mostrar al marshal. También cuentan con atributos relacionados con su comportamiento de *wander*, o el ángulo y distancia del cono de visión (para los aldeanos). *****

En cuanto al entorno, podemos encontrar las diferentes zonas, que almacenan el número de aldeanos actual, afectando al objetivo del agente, de tal forma que, si su objetivo es merodear en la zona, deberá decidir otra zona a la que transitar.

También se han creado tres estructuras de datos relacionadas con la dificultad elegida (fácil, media y difícil), que, respecto al comportamiento de los agentes, determinan la probabilidad de los aldeanos de mostrar información veraz o dudosa y el tiempo entre robos del ladrón. *****

*****En el apartado de **Fase de Testeo** se explican algunos de estos atributos más en profundidad.

## FASE DE TESTEO

Como bien se ha comentado anteriormente, *Marshallow: Pilferage in YolkTown* cuenta con tres niveles de dificultad: fácil, media y difícil. Dado que lo que se busca es el disfrute del jugador y que este pierda a veces, pero sin llegar a frustrarse, se ha buscado un equilibrio entre estas características para crear la mejor experiencia dependiendo de la dificultad que se elija. Por ejemplo, en la dificultad fácil el ladrón tendrá menos cuidado a la hora de robar (cumpliendo su objetivo igualmente, pero aun así "dejándose ganar" de forma más creíble que si simplemente se dejara atrapar), las pistas serán más propensas a ser veraces y el número de robos permitidos hasta que se acabe la partida es mayor, por tanto, para los

jugadores más pequeños y menos experimentados esta es la dificultad idónea. En cambio, en la mayor dificultad el ladrón tendrá los ojos más abiertos llevando mucho más cuidado, será más rápido efectuando robos, las pistas tenderán a ser dudosas y el número de robos permitidos será menor, siendo así un reto para el jugador experimentado, pero a la vez resultando totalmente posible salir victorioso, aunque la capacidad de memoria y contraste de las pistas por parte del usuario deberá ser muy superior en comparación a las otras dificultades, siendo a cambio una experiencia más satisfactoria cuando se gane la partida.

En resumen, se ha intentado tener en cuenta el principio K.I.S.S., ya que hemos creado una IA relativamente simple pero muy adaptable (según la dificultad) y entretenida a ojos del usuario.

A continuación, se mostrarán los parámetros a tener en cuenta para definir los distintos niveles de dificultad y cómo estos han sido modificados con cada sesión de testeo.

A la hora de testear el juego se han partido de unos parámetros base insertados por los programadores que se utilizaron durante el desarrollo principal del juego, de tal forma que se pudieran probar las funciones básicas del videojuego sin problema para solucionar cualquier bug que pudiese surgir. Si algunos de dichos parámetros no han sido modificados es debido a que cumplían con su funcionalidad a la perfección y no requerían de cambios para la mejora del comportamiento de los personajes y la IA.

Cabe comentar que el sistema de *wander* de los aldeanos se ha realizado mediante la *NavMesh*: el agente escoge, dentro de un área, una posición aleatoria de la *NavMesh* al azar que se encuentre dentro de la zona elegida por este. Tras ello se dirige a dicha posición y se queda quiero durante unos segundos (más detallado a continuación en el documento) mientras decide aleatoriamente mostrar un emoticono expresando su estado emocional, para después moverse a otro punto, repitiendo el proceso hasta que su tiempo en zona (especificado más adelante) termine.

Se han variado parámetros tanto de los agentes (que afectan a nivel general en todo el juego, independientemente de la dificultad en que se juegue) como de los modos de dificultad (únicos en cada nivel de dificultad que ofrece el videojuego, de tal forma que la experiencia varíe según el que se escoja):

## AGENTES

### LADRÓN

**Parámetros:**

- *Marshal Info Range*: distancia mínima entre el ladrón y el jugador que se tiene en cuenta a la hora de mostrar la información falsa.
- *MINIMUM_DESTINY_DISTANCE*: distancia mínima entre el ladrón y la zona que se tiene en cuenta a la hora de considerar que se encuentra en su destino.
- *Time To Change Zone*: tiempo que permanece el ladrón en la zona elegida una vez se encuentra en esta.
- *Still Time*: tiempo que el ladrón permanece quieto en la posición elegida dentro de la zona donde se encuentra.
- *WALKING_SPEED*: velocidad a la que se mueve el ladrón cuando camina.
- *RUNNING_SPEED*: velocidad a la que se mueve el ladrón cuando corre.
- *WANDER RADIUS*: área máxima que el agente tiene en cuenta a la hora de desplazarse entre posiciones cuando realiza el comportamiento *wander*.
- *Marshal Detect Range*: área que el ladrón tiene en cuenta para determinar si la distancia entre él y el jugador es demasiado pequeña como para arriesgarse a efectuar un robo.

- *STEALING_SPEED*: velocidad de movimiento del ladrón cuando está corriendo hacia su víctima.
- *MINIMUM_STEAL_DISTANCE*: distancia mínima a la que tiene que estar el ladrón de la víctima para efectuar un robo.
- *CLOSE_VILLAGERS_RANGE*: rango que el ladrón tiene en cuenta a la hora de escoger una víctima para, posteriormente, robarle.

Se han tocado los parámetros de las secciones “Parámetros” y “Parámetros LADRÓN” (que se encuentran en la ruta: Assets/Prefabs/Thief).

| Característica/Atributo | Fecha Modificación | Valor Antes | Valor Después |
|---|---|---|---|
| *Marshal Info Range* | 08/01/2021 | 5 | 7 |
| *MINIMUM_DESTINY_DISTANCE* | 08/01/2021 | 0.5 | 10 |
| *Time To Change Zone* | 08/01/2021 | 10 | - |
| *Still Time* | 08/01/2021 | 2.5 | - |
| *WALKING_SPEED* | 08/01/2021 | 2 | - |
| *RUNNING_SPEED* | 08/01/2021 | 4 | - |
| *WANDER_RADIUS* | 08/01/2021 | 8 | - |
| *Marshal Detect Range* | 08/01/2021 | 20 | 40 |
| *STEALING_SPEED* | 08/01/2021 | 10 | - |
| *MINIMUM_STEAL_DISTANCE* | 08/01/2021 | 2 | - |
| *CLOSE_VILLAGERS_RANGE* | 08/01/2021 | 10 | - |
| *Marshall Info Range* | 11/01/2021 | 7 | - |
| *MINIMUM_DESTINY_DISTANCE* | 11/01/2021 | 10 | - |
| *Time To Change Zone* | 11/01/2021 | 10 | - |
| *Still Time* | 11/01/2021 | 2.5 | - |
| *WALKING_SPEED* | 11/01/2021 | 2 | - |
| *RUNNING_SPEED* | 11/01/2021 | 4 | - |
| *WANDER_RADIUS* | 11/01/2021 | 8 | - |
| *Marshal Detect Range* | 11/01/2021 | 40 | 25 |
| *STEALING_SPEED* | 11/01/2021 | 10 | - |
| *MINIMUM_STEAL_DISTANCE* | 11/01/2021 | 2 | - |
| *CLOSE_VILLAGERS_RANGE* | 11/01/2021 | 10 | 16 |

## ALDEANO

**Parámetros:**

Tanto los aldeanos como el ladrón comparten los parámetros de la sección “Parámetros” de tal forma que así el ladrón pueda simular el comportamiento de todos los demás aldeanos para confundir al jugador. Por ello solo se explican los parámetros propios únicamente de los aldeanos:

- *Vision distance*: distancia máxima a la que los aldeanos presencian un robo mediante su cono de visión (para posteriormente hacer el papel de testigos).
- *Vision Angle*: ángulo máximo del cono de visión.

Se han tocado los parámetros de las secciones “Parámetros” y “Parámetros ALDEANO” (que se encuentran en la ruta: Assets/Prefabs/Villager).

| Característica/Atributo | Fecha Modificación | Valor Antes | Valor Después |
|---|---|---|---|
| *Marshal Info Range* | 08/01/2021 | 5 | 7 |
| *MINIMUM_DESTINY_DISTANCE* | 08/01/2021 | 0.5 | 10 |
| *Time To Change Zone* | 08/01/2021 | 10 | - |
| *Still Time* | 08/01/2021 | 2.5 | - |
| *WALKING_SPEED* | 08/01/2021 | 2 | - |
| *RUNNING_SPEED* | 08/01/2021 | 4 | - |
| *WANDER_RADIUS* | 08/01/2021 | 8 | - |
| *Vision distance* | 08/01/2021 | 7 | 15 |
| *Vision Angle* | 08/01/2021 | 120 | 180 |

## MARSHAL

**Parámetros:**

- *QUIT_STEAL_ICON_RANGE*: área con centro en el jugador que se tiene en cuenta a la hora de eliminar los iconos de robo de la interfaz si está suficientemente cerca de donde se cometió.
- *DETENTION_RANGE*: distancia a la que *Marshallow* se queda quieto frente al aldeano que se ha elegido para arrestar.
- *Nav Mesh_speed*: velocidad de movimiento de *Marshallow*.
- *Nav Mesh_acceleration*: aceleración de *Marshallow*.
- Nav Mesh_angularSpeed: velocidad angular de *Marshallow*.
- *Nav Mesh_stopping Distance*: determina la distancia a su destino a la que *Marshallow* frena.
- *Nav Mesh_auto Braking*: permite que se realice el freno al llegar al destino sin realizar deslizamientos.

Se han tocado los valores de las secciones “Rangos” y “Físicas *NavMesh*” (que se encuentran en la ruta: Assets/Prefabs/Player).

| Característica/Atributo | Fecha Modificación | Valor Antes | Valor Después |
|---|---|---|---|
| *QUIT_STEAL_ICON_RANGE* | 08/01/2021 | 10 | 20 |
| *DETENTION_RANGE* | 08/01/2021 | 2 | 4 |
| *Nav Mesh_speed* | 08/01/2021 | 8 | - |
| *Nav Mesh_acceleration* | 08/01/2021 | 12 | 24 |
| *Nav Mesh_angularSpeed* | 08/01/2021 | 5000 | 10000 |
| *Nav Mesh_stopping Distance* | 08/01/2021 | 0.2 | - |
| *Nav Mesh_auto Braking* | 08/01/2021 | true | - |

## MODOS DE DIFICULTAD

**Parámetros:**

- *Villagers*: número de aldeanos que aparecen por todo el pueblo.
- *Catch Attempts*: simboliza las "vidas" del jugador (es decir, el número de intentos de arresto erróneos permitidos antes de perder la partida).
- *Thief Robberies*: número de robos totales permitidos antes de perder la partida.
- *Victim Safe Probability*: probabilidad de que las víctimas proporcionen información segura (esto es, que los dos datos proporcionados sean verídicos al 100%).
- *Victim Veracity Probability*: probabilidad de que, si la víctima proporciona información dudosa, el dato que es dudoso sea realmente cierto.
- *Witness Safe Probability*: equivalente al parámetro de la víctima, pero para el testigo.
- *Witness Veracity Probability*: equivalente al parámetro de la víctima, pero para el testigo.
- *VILLAGER_SPEED_RUN_PROBABILITY*: probabilidad de que los aldeanos se desplacen entre zonas corriendo en vez de caminando.
- *Time Between Steals*: tiempo mínimo que transcurre entre un robo y el siguiente (se especifica mínimo ya que no se tiene en cuenta el tiempo que pasa cuando el ladrón está cambiando de zona).
- *THIEF_SPEED_RUN_PROBABILITY*: probabilidad de que el ladrón se desplace corriendo en vez de andando tras efectuar un robo (esto lo tenemos en cuenta para que se confunda entre los aldeanos que también corren, así se aumenta la dificultad a la hora de decidir quién es el ladrón).
- *fakeWitnessProbability*: probabilidad de que el ladrón se haga pasar por víctima.

Todos los parámetros se encuentran en la ruta: Assets/Resources/Difficulties/XXXX (refiriéndose con "XXXX" a cualquiera de las tres dificultades).

### FÁCIL

| Característica/Atributo | Fecha Modificación | Valor Antes | Valor Después |
|---|---|---|---|
| *Villagers* | 08/01/2021 | 30 | 20 |
| *Catch Attempts* | 08/01/2021 | 3 | - |
| *Thief Robberies* | 08/01/2021 | 8 | - |
| *Victim Safe Probability* | 08/01/2021 | 50 | 70 |
| *Victim Veracity Probability* | 08/01/2021 | 50 | 80 |
| *Witness Safe Probability* | 08/01/2021 | 50 | - |
| *Witness Veracity Probability* | 08/01/2021 | 50 | 70 |
| *VILLAGER_SPEED_RUN_PROBABILITY* | 08/01/2021 | 30 | 20 |
| *Time Between Steals* | 08/01/2021 | 30 | - |
| *THIEF_SPEED_RUN_PROBABILITY* | 08/01/2021 | 100 | - |
| *fakeWitnessProbability* | 08/01/2021 | 50 | - |
| *Villagers* | 11/01/2021 | 20 | - |
| *Catch Attempts* | 11/01/2021 | 3 | - |
| *Thief Robberies* | 11/01/2021 | 8 | 6 |
| *Victim Safe Probability* | 11/01/2021 | 70 | - |

| | | | |
|---|---|---|---|
| *Victim Veracity Probability* | 11/01/2021 | 80 | - |
| *Witness Safe Probability* | 11/01/2021 | 50 | - |
| *Witness Veracity Probability* | 11/01/2021 | 70 | - |
| *VILLAGER_SPEED_RUN_PROBABILITY* | 11/01/2021 | 20 | - |
| *Time Between Steals* | 11/01/2021 | 30 | 20 |
| *THIEF_SPEED_RUN_PROBABILITY* | 11/01/2021 | 100 | - |
| *fakeWitnessProbability* | 11/01/2021 | 50 | - |

MEDIA

| Característica/Atributo | Fecha Modificación | Valor Antes | Valor Después |
|---|---|---|---|
| *Villagers* | 08/01/2021 | 40 | 30 |
| *Catch Attempts* | 08/01/2021 | 2 | - |
| *Thief Robberies* | 08/01/2021 | 6 | 5 |
| *Victim Safe Probability* | 08/01/2021 | 50 | 40 |
| *Victim Veracity Probability* | 08/01/2021 | 50 | 60 |
| *Witness Safe Probability* | 08/01/2021 | 50 | 35 |
| *Witness Veracity Probability* | 08/01/2021 | 50 | - |
| *VILLAGER_SPEED_RUN_PROBABILITY* | 08/01/2021 | 30 | 50 |
| *Time Between Steals* | 08/01/2021 | 30 | 20 |
| *THIEF_SPEED_RUN_PROBABILITY* | 08/01/2021 | 100 | 65 |
| *fakeWitnessProbability* | 08/01/2021 | 50 | 20 |
| *Villagers* | 11/01/2021 | 30 | - |
| *Catch Attempts* | 11/01/2021 | 2 | - |
| *Thief Robberies* | 11/01/2021 | 5 | - |
| *Victim Safe Probability* | 11/01/2021 | 40 | 30 |
| *Victim Veracity Probability* | 11/01/2021 | 60 | - |
| *Witness Safe Probability* | 11/01/2021 | 35 | - |
| *Witness Veracity Probability* | 11/01/2021 | 50 | - |
| *VILLAGER_SPEED_RUN_PROBABILITY* | 11/01/2021 | 50 | - |
| *Time Between Steals* | 11/01/2021 | 20 | 15 |
| *THIEF_SPEED_RUN_PROBABILITY* | 11/01/2021 | 65 | - |
| *fakeWitnessProbability* | 11/01/2021 | 20 | - |

DIFÍCIL

| Característica/Atributo | Fecha Modificación | Valor Antes | Valor Después |
|---|---|---|---|
| *Villagers* | 08/01/2021 | 30 | 50 |

| | | | |
|---|---|---|---|
| ***Catch Attempts*** | 08/01/2021 | 1 | - |
| ***Thief Robberies*** | 08/01/2021 | 5 | - |
| ***Victim Safe Probability*** | 08/01/2021 | 50 | 30 |
| ***Victim Veracity Probability*** | 08/01/2021 | 50 | 35 |
| ***Witness Safe Probability*** | 08/01/2021 | 50 | 20 |
| ***Witness Veracity Probability*** | 08/01/2021 | 50 | 30 |
| ***VILLAGER_SPEED_RUN_PROBABILITY*** | 08/01/2021 | 30 | 50 |
| ***Time Between Steals*** | 08/01/2021 | 30 | 15 |
| ***THIEF_SPEED_RUN_PROBABILITY*** | 08/01/2021 | 100 | 50 |
| ***fakeWitnessProbability*** | 08/01/2021 | 50 | 5 |
| ***Villagers*** | 11/01/2021 | 50 | - |
| ***Catch Attempts*** | 11/01/2021 | 1 | - |
| ***Thief Robberies*** | 11/01/2021 | 5 | 4 |
| ***Victim Safe Probability*** | 11/01/2021 | 30 | - |
| ***Victim Veracity Probability*** | 11/01/2021 | 35 | - |
| ***Witness Safe Probability*** | 11/01/2021 | 20 | - |
| ***Witness Veracity Probability*** | 11/01/2021 | 30 | - |
| ***VILLAGER_SPEED_RUN_PROBABILITY*** | 11/01/2021 | 50 | - |
| ***Time Between Steals*** | 11/01/2021 | 15 | 10 |
| ***THIEF_SPEED_RUN_PROBABILITY*** | 11/01/2021 | 50 | - |
| ***fakeWitnessProbability*** | 11/01/2021 | 5 | - |

## REPARTO DE TAREAS

**Tareas de Programación:**

- **Esquema de los *Behaviour Trees:*** Samuel Ríos Carlos.
- **Implementación de nodos y métodos auxiliares:** Mario Belén Rivera, Mireya Funke Prieto y Samuel Ríos Carlos.
- **Programación de las mecánicas del juego:** Samuel Ríos Carlos.
- **Programación de interfaz:** Samuel Ríos Carlos.
- **Programación de *barks*:** Mario Belén Rivera y Samuel Ríos Carlos.
- **Implementación de audio:** Mireya Funke Prieto y Mario Belén Rivera.

**Tareas artísticas:**

- ***Level Design* (creación):** Mireya Funke Prieto.
- **Iluminación:** Mireya Funke Prieto y Mario Belén Rivera.
- **Diseño de personajes:** Enrique Sánchez de Francisco.
- **Modelado, texturizado y animación de Personajes:** Sergio Cruz Serrano.
- **Diseño, modelado y texturizado de escenarios:** Mireya Funke Prieto.
- **Diseño de VFX:** Mario Belén Rivera.

**Diseño de juego:**

- ***Level Design* (redacción):** Sergio Cruz Serrano y Enrique Sánchez de Francisco.

- **Diseño de comportamiento de los agentes:** Enrique Sánchez de Francisco y Sergio Cruz Serrano.
- **Encargados del Testeo:** Enrique Sánchez de Francisco, Sergio Cruz Serrano, Mario Belén Rivera y Samuel Ríos Carlos.
- **Diseño de *Behaviour Trees*:** Sergio Cruz Serrano.

**Gestión del grupo de trabajo:**

- **Organización de roles y tareas:** ED (equipo de desarrollo).
- **Propuesta de la metodología de trabajo:** Enrique Sánchez de Francisco.

## LECCIONES APRENDIDAS

A lo largo del desarrollo del proyecto nos hemos dado cuenta de la importancia de herramientas como *Trello* o *Github* a la hora de realizar un reparto equitativo de tareas, dejar claras las dependencias entre estas y, sobre todo, mantener un trabajo organizado y bien coordinado. Creemos que es una buena práctica que ha ayudado a llevar un correcto desarrollo y que tendremos en cuenta para futuros proyectos. Otro aspecto que hemos llevado a cabo y que consideramos muy importante para un correcto desarrollo, es el de sentar las bases de los comportamientos gracias a el diseño y la especificación de requisitos de estos antes de pasar a su implementación. Por ejemplo, una vez ya estaban hechos el esquema de los árboles con sus respectivas acciones y condiciones, resultó mucho más sencilla y directa su traducción a código.

Por otro lado, comentar que, aunque era nuestra primera vez utilizando árboles de decisión, nos ha resultado una estructura realmente versátil y cómoda, de tal forma que hemos podido llevar a cabo la idea que queríamos desarrollar sin problema. Por ejemplo, nos ha resultado muy conveniente el poder reutilizar nodos entre distintos agentes y el uso de los decoradores, reduciendo el número total de nodos.

También nos gustaría destacar que, para llevar a cabo una buena inteligencia artificial, no solo basta con implementar la idea, si no que se debe pulir su comportamiento realizando pruebas de testeo. Gracias a las pruebas realizadas por el equipo de desarrollo y la retroalimentación de usuarios externos, hemos podido ir refinando los atributos de nuestros agentes para obtener un resultado entretenido y satisfactorio.

Sin embargo, comentar que nos habría gustado que se hubiesen generado comportamientos emergentes para aportar variedad a la jugabilidad. Además, creemos que deberíamos pulir más la forma en la que interactúan los agentes entre sí. A futuro nos gustaría repasar este último aspecto obteniendo un resultado más variado y realista, por ejemplo, que no se obstruyan el camino entre ellos, que realicen interacciones de ocio o que interactúen entre ellos.

## LICENCIAS

## REFERENCIAS BIBLIOGRÁFICAS

- Vídeo seguido para implementar la base de los *Behaviour Trees*: https://www.youtube.com/watch?v=F-3nxJ2ANXg&ab_channel=GameDevChef

- Fuentes usadas en el proyecto:

https://www.dafont.com/dimbo.font

https://www.dafont.com/goodlight.font