COMPORTAMIENTO DE PERSONAJES – GRUPO 9

Vida en un castillo medieval

Juan Pedro Guirado Sánchez
Ángel Luis Serrano González

Contenido

Descripción general

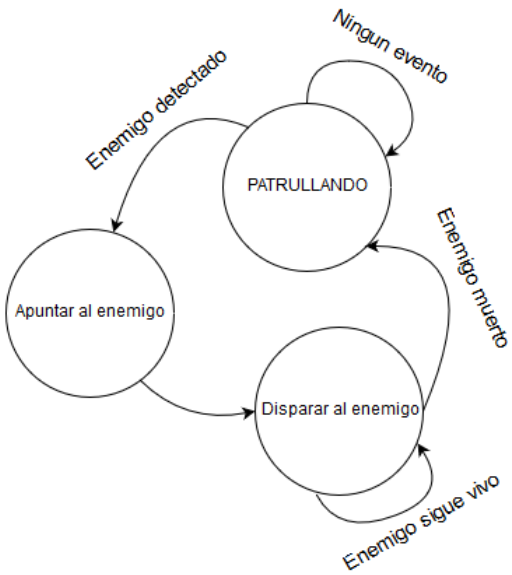
El objetivo de esta práctica es partir del escenario desarrollado para la asignatura de Animación 3D para programar ciertos personajes, con sus determinados comportamientos, dentro del castillo para darle vida al mismo.

La práctica no es interactiva. Sólo se podrá manejar la cámara para observar lo que está sucediendo en el escenario. Con el ratón se controla el paneo de la cámara y el movimiento de esta se realizará con las teclas WASD. Lo que si podrá realizar el espectador es producir la generación forzada de aparición de fuego o enemigos con las teclas F y E respectivamente. Por otro lado, la práctica se desarrolla en un entorno 3D, basándose en el modelo de castillo desarrollado para la asignatura de Animación 3D.

En lo que se refiere a los agentes, en nuestra práctica hemos desarrollado 5 tipos de personajes y 2 elementos que provocarán las transiciones de estados de los personajes. Estos elementos son: fuego y enemigo. En cuanto a los personajes, en primer lugar, hablaremos de los arqueros. Estos están encargados de patrullar las murallas del castillo y si se produce la aparición de un enemigo, procederían a dispararle hasta eliminarlo. En según lugar, los caballeros se encargan de patrullar en el interior del castillo y vigilar. Si se produjese la aparición de un fuego, estos procederían a apagarlo. En tercer lugar se encuentran el rey y su guardia, aunque son personajes diferentes. El rey se encuentra normalmente paseando por la placeta del santuario y la guardia se encuentra en reposo debajo de las escaleras de esta placeta. Si se produce la aparición de un enemigo, el rey y la guardia son avisados y proceden a realizar una formación conjunta de defensa, situándose el rey en el centro y la guardia haciendo un circulo alrededor de él para protegerlo. En último lugar se encuentran las campesinas, que normalmente se encuentran o en reposo o paseando por el castillo. Si se produce la aparición de un fuego, todas huyen a la puerta del castillo por temor al fuego.

Descripciones detalladas de los personajes

Arqueros



Descripción textual detallada

Los arqueros están encargados de patrullar las murallas para defender al castillo, su gente y el rey del ataque de los barbaros enemigos. Normalmente se encuentran patrullando. Si se produce la aparición de un enemigo, el arquero más cercano a el comenzará a dispararle hasta eliminarlo. Una vez eliminado, volverá a su patrulla habitual.

Tabla de percepciones

Las percepciones han sido programadas dentro del código de la propia FSM

Tabla de acciones

void mirarEnemigo()

El arquero apunta hacia el enemigo que ha aparecido

private IEnumerator DispararEnemigo()

Espera 10 segundos y entre cada uno de esos segundos, realiza 10 de daño al enemigo simulando que le está disparando.

public void setDestino(Vector3 nuevoDestino)

Dirige al arquero a la posición indicada en nuevoDestino.

public void disparar()

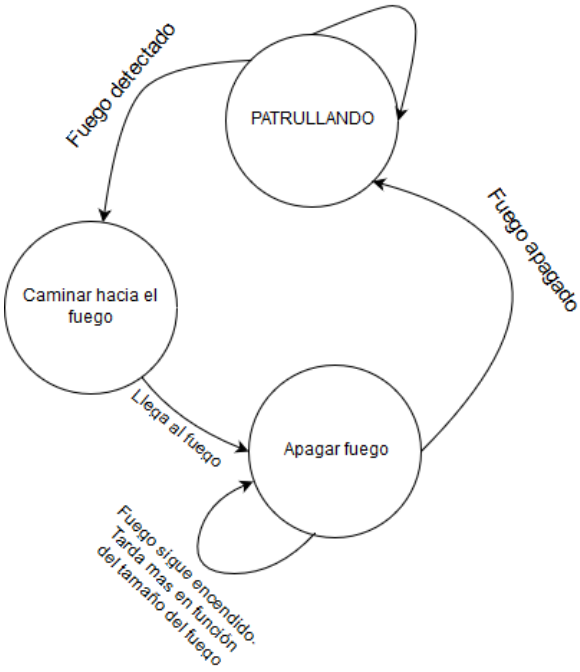
Activa la animación de disparo de flechas.

public void enemigoMuerto()

Desactiva la animación de disparo.

El resto de las acciones se han programado dentro del propio código de la FSM

Caballeros



Descripción textual detallada

Los caballeros están encargados de patrullar los interiores del castillo. Normalmente se encuentran patrullando. Si se produce la aparición de un fuego, el caballero mas cercano al mismo se acercará a apagarlo. El tamaño del fuego puede variar de forma aleatoria. El tiempo que tardará el caballero en apagar este fuego dependerá del tamaño del fuego. Una vez apagado el fuego, volverán a su patrulla normal.

Tabla de percepciones

Las percepciones han sido programadas dentro del código de la propia FSM

Tabla de acciones

public void setDestino(Vector3 nuevoDestino)

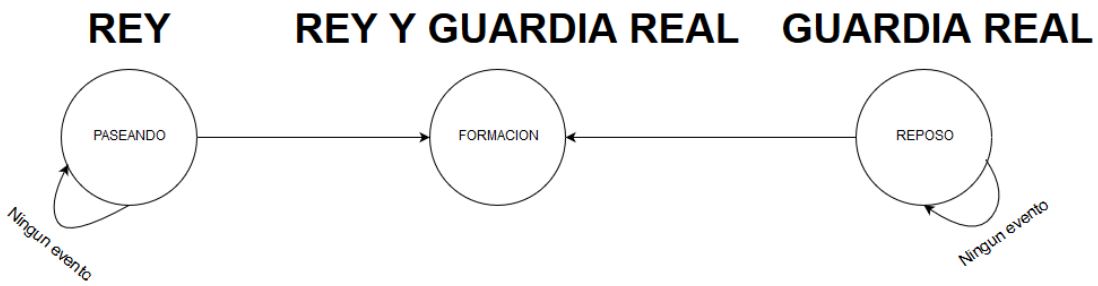
Dirige al caballero a la posición indicada en nuevoDestino.

private IEnumerator apagarFuego()

Espera una cantidad x de segundos que dependerá del tamaño del fuego para simular que el caballero está apagando el fuego.

El resto de las acciones se han programado dentro del propio código de la FSM

Rey



Descripción textual detallada

El rey se encuentra normalmente paseando por la placeta del santuario del castillo. Si se produce la aparición de un enemigo, se alerta al rey y a su guardia. Estos realizan una formación conjunta donde el rey se sitúa en el centro de la formación y su guardia se sitúa en círculo alrededor de él. Una vez la amenaza ha desaparecido, el rey vuelve a su paseo habitual.

Tabla de percepciones

Las percepciones han sido programadas dentro del código de la propia FSM.

Tabla de acciones

public void setDestino(Vector3 nuevoDestino)

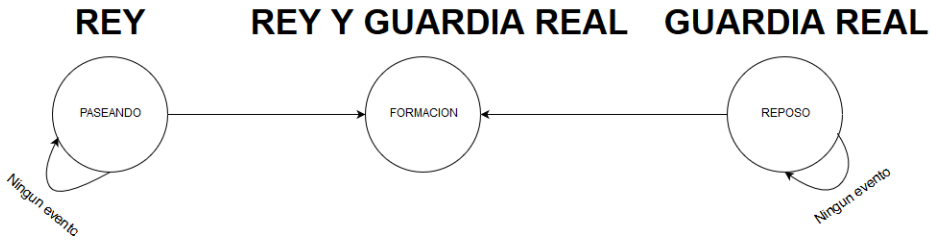
Dirige al rey a la posición indicada en nuevoDestino.

void controlAnimaciones()

Controla las animaciones en función de si el personaje está parado o no.

El resto de las acciones se han programado dentro del propio código de la FSM

Guardia Real



Descripción textual detallada

La guardia real normalmente se encuentra en reposo a los pies de la escalera de la placeta donde se encuentra paseando el rey. Si se produce la aparición de un enemigo, se alerta al rey y a su guardia. Estos realizan una formación conjunta donde el rey se sitúa en el centro de la formación y su guardia se sitúa en círculo alrededor de él.

Tabla de percepciones

Las percepciones han sido programadas dentro del código de la propia FSM.

Tabla de acciones

public void setDestino(Vector3 nuevoDestino)

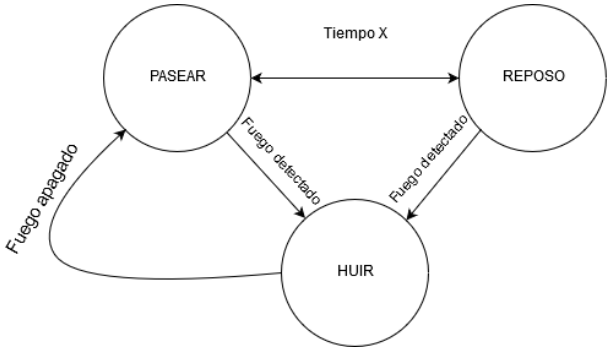
Dirige al guarda real a la posición indicada en nuevoDestino.

void controlAnimaciones()

Controla las animaciones en función de si el personaje está parado o no.

El resto de las acciones se han programado dentro del propio código de la FSM

Campesina



Descripción textual detallada

Las campesinas se encuentran normalmente o en reposo o paseando por los interiores del castillo. Si se produce la aparición de un fuego, las campesinas correrán aterrorizadas al exterior del castillo para evitar al fuego. Una vez el fuego ha sido extinguido, estas vuelven a su comportamiento habitual dentro de las murallas

Tabla de percepciones

Las percepciones han sido programadas dentro del código de la propia FSM.

Tabla de acciones

public void setDestino(Vector3 nuevoDestino)

Dirige a la campesina a la posición indicada en nuevoDestino.

void controlAnimaciones()

Controla las animaciones en función de si el personaje está parado o no.

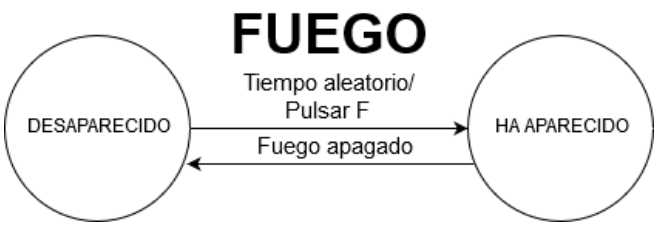
private IEnumerator TiempoCambioReposoPaseando()

Controla el cambio de reposo a paseando en función de un determinado tiempo.

El resto de las acciones se han programado dentro del propio código de la FSM

Elementos

Fuego



Descripción textual detallada

El fuego aparece después de un tiempo aleatorio entre 2 intervalos o pulsando el botón F si se ha activado esta característica en el script del fuego llamada “Generacion Fuego Teclado”. Una vez aparece, su tamaño también será aleatorio entre 2 intervalos modificando su escala. En función de este tamaño, el caballero tardará más o menos en apagar el fuego.

Tabla de percepciones

public bool HayFuego()

Indica si hay fuego o no en este momento.

public Vector3 GetPosicionFuego()

Devuelve la posición del fuego.

Tabla de acciones

private IEnumerator GeneradorFuego()

Espera un tiempo aleatorio entre dos valores determinados y genera el fuego llamando a HazVisible()

public void HazInvisible()

Hace desaparecer el fuego

public void HazVisible()

Hace aparecer un fuego en unas de las posiciones que se le han puesto. El tamaño del fuego se establece de forma aleatoria entre dos valores determinados.

public void ApagaFuego()

Apaga el fuego llamando a HazInvisible y elige una nueva posición para el siguiente fuego.

Enemigo

Descripción textual detallada

El enemigo aparece después de un tiempo aleatorio entre 2 intervalos o pulsando el botón E si se ha activado esta característica en el script del fuego llamada “Generacion Enemigo Teclado”. Una vez aparece, se dirigirá hacia el interior del castillo. Los arqueros se encargan de eliminarlo. Una vez es eliminado, volverá a aparecer después del tiempo aleatorio.

Tabla de percepciones

public bool HayEnemigo()

Indica si hay un enemigo o no en este momento.

public Vector3 GetPosicionEnemigo()

Devuelve la posición del enemigo.

Tabla de acciones

private IEnumerator GeneradorEnemigo()

Espera un tiempo aleatorio entre dos valores determinados y genera el enemigo, en una de las posiciones que se le han asignado, llamando a HazVisible()

public void HazInvisible()

Hace desaparecer al enemigo una vez ha sido derrotado

public void HazVisible()

Hace aparecer un enemigo en unas de las posiciones que se le han puesto.

public void ApagaFuego()

Apaga el fuego llamando a HazInvisible y elige una nueva posición para el siguiente fuego.

public void TakeDamage(int amount)

El enemigo recibe una cantidad de daño determinada por amount y se resta a su vida actual.

public void EnemigoDerrotado()

Se llama cuando la salud del enemigo es 0, hace desaparecer al enemigo llamando a HazInvisible. Establece una nueva posición de aparición del enemigo y restablece su salud al máximo.

Flujo y estructuras de datos

Como fluye la información desde y hacia el entorno

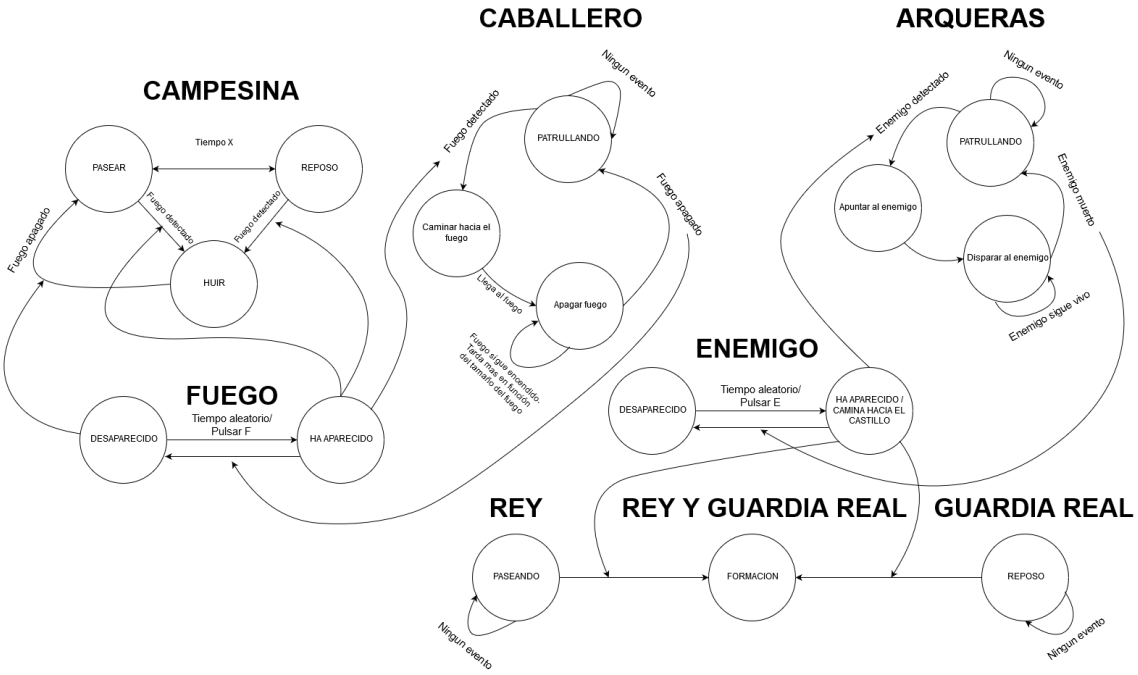


Ilustración 1: Se incluye esta imagen en los adjuntos

Como fluye la información entre personajes.

Los personajes que dependan de un elemento (fuego, enemigo) incluye en su script una referencia al game object de estos elementos. Una vez tenemos el GameObject accedemos al script encargado de controlar el comportamiento, con el cual recibiremos la información necesaria para realizar la transición entre estados, tipo Pull. A su vez, si un personaje necesita información de otro lo hemos realizado de la misma forma, incluyendo una referencia a su GameObject.

Estructuras de datos relevantes para el comportamiento de los personajes

Arquero

Variable	Uso
GameObject enemigoObjetivo	Poder acceder al script que controla al enemigo y que nos dé información para las transiciones de estados
GameObject [] arqueros	GameObject de los arqueros disponibles para controlar quien dispara al enemigo en función de la distancia
Enum Testado {PATRULLANDO,APUNTARENEMIGO,MATARENEMIGO};	Enumerado para controlar los estados

TEstado estado	Estado actual del personaje
ControlMovimientoArqueros movArquero;	Variable con la que accederemos al script de movimiento del arquero encargado de matar al enemigo
ControladorEnemigo controladorObjetivo	Variable en la que guardaremos la referencia al script que controla al enemigo con la que recibiremos información para la transición de estados
ControlArqueros esteArquero	Referencia al script del arquero que se encargará de disparar al enemigo para que se produzca de forma correcta la transición de estados entre todas las instancias de arqueros

Caballero

Variable	Uso
GameObject fuegoObjetivo	Poder acceder al script que controla al fuego y que nos dé información para las transiciones de estados
GameObject [] caballeros	GameObject de los caballeros disponibles para controlar quien acude a apagar el fuego
enum TEstado { PATRULLANDO, CAMINARFUEGO, APAGANDOFUEGO }	Enumerado para controlar los estados
TEstado estado	Estado actual del personaje
ControlMovimientoArqueros movCaballero	Variable con la que accederemos al script de movimiento del caballero encargado de apagar el fuego
ControladorFuego controladorObjetivo	Variable en la que guardaremos la referencia al script que controla el fuego con la que recibiremos información para la transición de estados

Rey

Variable	Uso
GameObject enemigoObjetivo	Poder acceder al script que controla al enemigo y que nos dé información para las transiciones de estados
GameObject [] caballeros	GameObject de los caballeros disponibles para controlar quien acude a apagar el fuego

enum TEstado {PASEANDO,FORMACION}	Enumerado para controlar los estados
TEstado estado	Estado actual del personaje
ControladorEnemigo controladorObjetivo	Variable en la que guardaremos la referencia al script que controla al enemigo con la que recibiremos información para la transición de estados

Guardia Real

Variable	Uso
GameObject enemigoObjetivo	Poder acceder al script que controla al enemigo y que nos dé información para las transiciones de estados
GameObject reyObjetivo;	Poder acceder al script que controlar al Rey para obtener datos importantes como su posición
enum TEstado { REPOSO, FORMACION}	Enumerado para controlar los estados
TEstado estado	Estado actual del personaje

Campesina

Variable	Uso
GameObject fuegoObjetivo	Poder acceder al script que controla al fuego y que nos dé información para las transiciones de estados
ControladorFuego controladorObjetivo;	Variable en la que guardaremos la referencia al script que controla e fuego con la que recibiremos información para la transición de estados
enum TEstado {REPOSO,PASEANDO,HUYENDO}	Enumerado para controlar los estados
TEstado estado	Estado actual del personaje

Enemigo

Variable	Uso
Vector3[] posiciones	Vector que almacena las posiciones donde puede aparecer el enemigo. Se pueden añadir todas las deseadas

bool hayEnemigo	Variable que nos ayudará a saber si hay un enemigo vivo en ese momento
int currentHealth	La vida actual del enemigo
Vector3 posicionAtaque	Posición a la que se dirigirá el enemigo cuando aparezca
bool generaciónEnemigoTeclado	Variable booleana que, si es activada, nos permitirá generar un enemigo pulsando la tecla E

Fuego

Variable	Uso
Vector3[] posiciones	Vector que almacena las posiciones donde puede aparecer el fuego. Se pueden añadir todas las deseadas
bool hayFuego	Variable que nos ayudará a saber si hay un fuego activo en ese momento
bool generacionFuegoTeclado;	Variable booleana que, si es activada, nos permitirá generar un fuego pulsando la tecla F
Vector3 tamañofuego	Vector3 que usaremos para cambiar la escala del fuego para cambiarlo de tamaño

Descripción de algoritmos

NavMeshes

En esta parte queríamos comentar que para realizar los distintos movimientos que los personajes hacen por el mapa hemos optado por la utilización de **NavMeshes**. Nos ha parecido una herramienta muy potente y sencilla, la cual nos ha permitido generar sin apenas retoques las zonas accesibles por las que se pueden mover nuestros personajes.

Tuvimos una serie de problemas con la posición virtual que genera el agente de NavMeshes con la que va actualizando la posición real del GameObject al que va atado, aunque pudimos resolverlo con éxito con la ayuda de la función Warp que incluyen estos NavMeshes. Cada uno de los personajes lleva asignadas las distintas posiciones por las que se mueve en el mapa. Haciendo uso de la función SetDestination con las posiciones asignadas conseguimos que los personajes realizarán los movimientos que esperábamos de ellos.

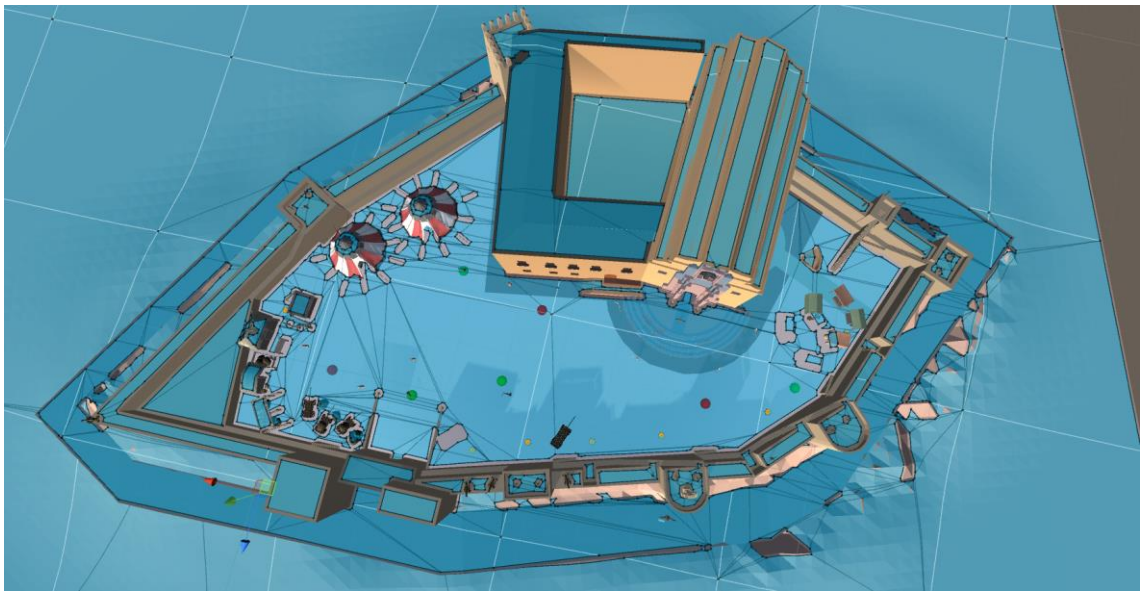


Ilustración 2: Esta imagen se encuentra en los adjuntos

Reparto de tareas

En lo que se refiere al reparto de tareas hemos optado por una división entre el apartado visual y técnico. Ángel Luis Serrano González ha sido el encargado de preparar todos los personajes y asignarles sus animaciones y componentes necesarios para su correspondiente funcionamiento. A su vez, ha sido el encargado de preparar el escenario y añadirle al mismo distintos assets para darle un aspecto más vivo. Por el otro lado, Juan Pedro Guirado Sánchez ha sido el encargado de programar todo lo que se refiere al comportamiento de los personajes, añadiendo scripts a los mismos para poder controlar sus estados y movimientos para que se realizaran de forma correcta. A su vez, también ha sido el encargado de diseño del juego y de los distintos tipos de personajes que aparecerían en él y como se comportarían.