Departamento de Ingeniería del Software e Inteligencia Artificial
Facultad de Informática
Universidad Complutense de Madrid

Práctica 2: El hilo de Ariadna

Revisión de documentación, resultados y final: 3, 10 y 13 de marzo de 2022

Importante: Las revisiones y calificaciones se realizan sobre *el grupo en su conjunto*, en tiempo y forma, asumiendo que se documente y acredite un reparto justo del trabajo. Los profesores deben ser los únicos con acceso a vuestro repositorio P2 dentro de la organización de GitHub IAV22-G*XX*, siendo *XX* vuestro número de grupo con dos dígitos. En el repositorio debe encontrarse la documentación, el ejecutable y el código fuente del proyecto. La documentación *README.md* debe estar en la raíz del repositorio y contener vuestros datos, un resumen del enunciado, la descripción del punto de partida proporcionado, el diseño de la solución (prototipo explicado con diagramas y/o pseudocódigo) y el enlace a un video oculto de YouTube titulado IAV22-G*XX*-P2 con la grabación personalizada de todas las *pruebas realizadas*, convenientemente rotuladas y comentadas, de menos de *3 minutos* de duración. El ejecutable para Windows de 64bits IAV22-G*XX*-P2.zip debe estar publicado como *lanzamiento* descargable en el repositorio. El código fuente debe incluir en carpetas separadas cualquier recurso o *plugin* de terceros que sea necesario para la correcta compilación del proyecto.

1. Introducción

"El segundo canto de El Hilo de Ariadna narra las aventuras de Teseo y su llegada a la isla de Creta, en busca de la cabeza del Minotauro. Ariadna, decidida a salvar al famoso héroe, le entregó un ovillo de hilo para que el laberinto perdiera por completo su secreto, y así Teseo pudiera escapar vivo de la isla. La princesa cretense empezaba a enamorarse perdidamente de él, y este le ofreció todo tipo de promesas que un hombre puede hacerle a una mujer.

El destino de Teseo estaba escrito, mató al Minotauro y se llevó consigo a Ariadna hasta la isla de Naxos, en la cual se celebró hasta el amanecer la gloria del héroe. A la mañana siguiente, Teseo abandonó a su prometida, mientras ella dormía en la playa. Por eso el canto termina con la maldición que Ariadna realiza sobre su amado..."

El conocido mito griego (Figura 1) nos sirve como excusa para profundizar ahora en el problema de la navegación de entornos virtuales, tan habitual en el mundo de los videojuegos. En la historia tenemos un protagonista, Teseo, en principio heróico. Hay también un villano, el temido Minotauro. Tenemos también el propio entorno donde se encuentran ambos personajes, el laberinto del Minotauro, y por último, el hilo mágico que Ariadna entregó a Teseo. Este hilo tiene la propiedad de burlar el embrujo del laberinto y ayudarnos a salir de él. Aunque en el mito se habla de la victoria de Teseo sobre el Minotauro, nosotros vamos a aplicarle la "maldición" que le lanza Ariadna de forma anticipada, y haremos que sea imposible que pueda acabar con la dichosa criatura. Lo único que hará el héroe cuando se encuentre próximo al monstruo es entablar combate, lo que en la práctica supone ralentizar su movimiento.

Figura 1. Óleo sobre lienzo de Niccolò Bambini donde la princesa Ariadna entrega su famoso ovillo de hilo al héroe ateniense Teseo, bajo la tutela de la diosa Atenea.

En esta ocasión, crearemos un prototipo en el que el jugador controla el movimiento de Teseo por los pasillos del laberinto. La bestia mitológica por su parte será un agente inteligente que se dedica a merodear por allí, siguiendo a Teseo en caso de percibirlo. El hilo de Ariadna funcionará, cuando lo activemos, dibujando con una línea blanca el camino de menor coste que puede conducir a Teseo hasta la *baldosa de salida*. El hilo, al ser mágico, tiene en cuenta todos los costes, ya que la baldosa donde se encuentra el Minotauro es intransitable, y moverse a las 9 baldosas vecinas tienen 5 veces mayor coste que el movimiento a través de una baldosa normal.

Este prototipo aplicará el algoritmo A* como técnica de navegación (búsqueda de caminos con información heurística), muy usada en videojuegos, en combinación con comportamientos de dirección como Merodear o Perseguir, y con el algoritmo de suavizado de caminos.

2. Planteamiento del proyecto

Desarrolla un prototipo de IA para Videojuegos, dentro de un entorno virtual laberíntico, con un agente inteligente que merodea por allí y un avatar, en principio controlado por el jugador. Al activar el hilo de Ariadna Teseo pasará a ser controlado por la máquina y procederá a salir del laberinto de manera tranquila pero automática, hasta que lo desactivemos. El prototipo será fácilmente usable y se basará en un entorno de *tamaño configurable*, con habitaciones y pasillos, incluyendo algunos muy estrechos, donde estarán el avatar y el enemigo.

Figura 2. Ejemplo de representación del laberinto del Minotauro.

La **documentación** del prototipo deberá ser la adecuada (datos correctos, resumen completo del enunciado, descripción del punto de partida con suficiente detalle, investigación de posibles referentes, diseño de la solución mediante diagramas y/o pseudocódigo…) **[2,5 ptos.]**.

El **ejecutable** será práctico y fácil de usar, comprendiendo -con la calidad mínima esperable en el prototipo de un videojuego- las siguientes funcionalidades:

A. Mostrar el entorno virtual (el laberinto del Minotauro) de tamaño *configurable*, con un esquema de división de *grafo de baldosas* que incluirá una baldosa de salida, donde se ubica inicialmente el avatar (Teseo). Debe haber varios caminos alternativos para llegar a la salida, algunos más anchos y otros muy estrechos (pasillos de una única baldosa de anchura). El avatar estará controlado por el jugador mediante los *cursores* **[0,5 ptos.]**.
B. Situar en el centro del laberinto al agente inteligente que representa al enemigo (el Minotauro), quien realizará un merodeo constante, pasando a perseguir al avatar si se lo encuentra en su línea de visión **[0,5 ptos.]**.
C. Representar el hilo de Ariadna (camino más corto a la baldosa de salida) pintado con una línea blanca y destacando las baldosas también con círculos blancos, a la vez que se activa la navegación automática de Teseo hasta la salida, todo ello mientras se mantenga pulsada la *barra espaciadora* **[1 pto.]**.
D. Elegir (activando o desactivando la funcionalidad con la tecla *S*) si *suavizar* o no el camino generado por el algoritmo anterior, generalmente reduciendo las baldosas que forman parte del camino suavizado a la salida **[1 pto.]**.
E. Desarrollar el movimiento completo de Teseo, que mientras tenemos pulsada la *barra espaciadora*, va moviéndose automáticamente siguiendo el hilo hacia la baldosa de salida. Esto hace desaparecer la parte del hilo que ya se ha recorrido, hilo que desaparece al completo en cuanto se suelta la barra espaciadora y se vuelve al movimiento manual **[1 pto.]**.

El *banco de pruebas* realizadas y documentadas en la grabación, cubrirá todas las funcionalidades del prototipo **[1 pto.]**, demostrando sus posibilidades con aquellas *métricas* que permitan valorar la eficiencia de la implementación: en este caso, tamaño del entorno en número

de casillas totales, casillas exploradas, longitud y coste del camino encontrado, así como el tiempo (ms) tardado en encontrarlo **[1 pto.]**.

El **código fuente** y las buenas prácticas de desarrollo software (*commits* frecuentes en la rama principal, organización de recursos en el proyecto y de objetos en la escena, escritura sistemática de los comentarios, etc.) se tendrán en cuenta en la revisión final **[1,5 ptos.]**.

3. Restricciones y consejos

A la hora de desarrollar este proyecto es obligatorio:

- Utilizar únicamente las herramientas de Unity y opcionalmente los *plugins* de terceros acordados con el profesor, sin reutilizar código ajeno a estos.
- Documentar claramente los algoritmos, heurísticas o cualquier "truco" utilizado.
- Diseñar y programar de la manera más limpia y elegante posible, separando la parte visual e interactiva del juego, del modelo y las técnicas de IA implementadas.
- Evitar, en la medida de lo posible, el uso de recursos audiovisuales pesados o ajenos.

Para realizar las pruebas y facilitar las revisiones de los profesores, intentando aprovechar el esfuerzo de desarrollo, conviene crear una interfaz gráfica cómoda para mostrar distintos escenarios de ejemplo, instrucciones de uso, etc. Por ejemplo, habilitando una "consola de trucos" (o teclas rápidas) que permitan ver los datos de los desarrolladores, reiniciar la ejecución, cambiar la cámara, establecer situaciones específicas para hacer pruebas, hacer invencible al avatar, etc. El manejo debe ser ágil e intuitivo para poder realizar rápidamente todas las pruebas con aquellas variaciones que puedan resultar interesantes.

4. Referencias y ampliaciones

Como base para vuestra investigación, además de la bibliografía de la asignatura, puedes aprovechar las siguientes referencias. En ningún caso debes replicar código que encuentres por ahí; asegúrate de entenderlo y verifica que funciona *exactamente* como se pide que lo haga.

- Unity 2018 Artificial Intelligence Cookbook, Second Edition (Repositorio)
  https://github.com/PacktPublishing/Unity-2018-Artificial-Intelligence-Cookbook-Second-Edition
- Unity Artificial Intelligence Programming, Fourth Edition (Repositorio)
  https://github.com/PacktPublishing/Unity-Artificial-Intelligence-Programming-Fourth-Edition

Para ir más lejos en tu aprendizaje y alcanzar la máxima calificación, realiza estas ampliaciones:

- Genera procedimentalmente el entorno con verdadera forma de laberinto, utilizando algoritmos específicos para ello.
- Modifica al Minotauro para que patrulle siguiendo un camino con un patrón concreto, y de paso permite tener un número variable de monstruos en el laberinto.
- Añade zonas de baldosas con distinto coste al laberinto, como agua, barro, pendientes…
- Da la opción de poder cambiar la heurística utilizada en el algoritmo A*.
- Permite añadir más salidas al laberinto y modifica a Teseo para que, si hay varias salidas, salga por la más cercana, utilizando para ello el algoritmo de Dijkstra.