PROGRAMACION DE JUEGOS EN UNITY

Introducción

El objetivo de la práctica es la aplicación de los conceptos del paradigma de programación orientado a objetos para **programar un videojuego en 3D que implemente algún tipo de técnica de IA en su desarrollo, utilizando para su creación el motor gráfico de juegos Unity** (http://unity3d.com/es).

Formas de programar IA en los juegos

Como sabemos, los primeros videojuegos carecían de IA de forma que los personajes se comportaban de una manera fija (realizando desplazamientos según una ruta predeterminada) o al azar (realizando desplazamientos, disparos, etc. de forma aleatoria), lo cual provocaba que el juego resultase monótono y predecible llegando a aburrir al jugador dado que éste no se sentía retado por el propio juego.

Para resolver dichos problemas se empezaron a utilizar diferentes técnicas de IA (cada una con una aplicabilidad específica) en la programación de los videojuegos consiguiéndose hacer estos más atractivos de cara al jugador al conseguir un comportamiento "inteligente" de los personajes que intervenían en el juego.

Como vimos en la práctica anterior, las primeras técnicas de IA que se usaron en los videojuegos fueron la **Búsqueda de Caminos o Pathfinding** (para dar movimiento inteligente a los personajes determinando en qué dirección se ha de mover a través de la escena para llegar al destino deseado desde una determinada posición), los **Árboles de Decisión o Decision Tree** (generalmente utilizado en los videojuegos tanto para tomar decisiones en base a la información de que dispone como para el control de animaciones), **Maquinas de Estados o Finite State Machines** (para generar una acción o comportamiento -huir, perseguir, esconderse, etc.- en función del estado en el que el personaje se encuentre, e indicar las condiciones -nos disparan, trascurre un cierto tiempo, etc.- para transitar hacia otra estado) y **Árboles de Comportamiento o Behavior Trees** (herramienta muy popular para programar la IA de los personajes y el comportamiento de los agentes gracias a que permiten hacer más sencilla la inclusión de nuevos comportamientos, añadiendo complejidad a las respuestas de los personajes).

Además de las técnicas enumeradas anteriormente, existen otras técnicas más recientes y complejas que también pueden ser aplicadas tales como los Algoritmos Genéticos, Redes Neuronales, Lógica Difusa.

LÓGICA DIFUSA (Fuzzy Logic)

La Lógica Difusa permite modelar conocimiento impreciso y cuantitativo, así como transmitir y manejar incertidumbre y soportar el razonamiento humano de una forma natural y se introdujo en videojuegos (de forma fehaciente) en 1996, según la revista Game Developer Magazine de Larry O'Brein.

La lógica difusa se puede considerar una extensión de la lógica clásica, en la que se incorporan nuevos conceptos para trabajar con valores imprecisos. La diferencia fundamental entre la lógica difusa y la clásica está en el rango de valores de verdad que se pueden dar.

Mientras que en las proposiciones clásicas sólo existen dos posibles valores de verdad (verdadero o falso), el grado de verdad o falsedad de las proposiciones difusas puede tomar distintos valores numéricos, de forma que permite representar mejor el mundo real (donde raramente las cosas son verdaderas o falsas, o las medidas tienen un grado de precisión absoluta), al permitir trabajar tanto con valores de entrada como de salida aproximados: no precisos

La lógica fuzzy se usa principalmente (aunque no sólo) para implementar Sistemas de control y regresión, Sistemas de clasificación y Algoritmos de agrupamiento (clustering).

En cuanto a su aplicación a los videojuegos, la lógica fuzzy puede ser:

- Un mecanismo de decisión: por ejemplo, un controlador para la selección de recursos o armas (NPC decision making) en función de diferentes parámetros (antecedentes distancia al objetivo y estado de la munición, consecuente conveniencia o idoneidad del objetivo).

- Un mecanismo de clasificación: para gestionar el nivel de destreza de jugadores activando elementos de un personaje en función de la evaluación del oponente (por su fuerza, armas, defensas o categoría, dificultad, etc.), para evaluar las amenazas (ej: Halo 3) dimensionando la fuerza defensiva de forma más humana y menos predecible.

- Un mecanismo de control: para movimiento en general (control del rumbo de un vehículo, movimientos más suaves y naturales en vehículos o personajes, etc.) o para fusionar percepciones, acciones y/o comportamientos.

ALGORITMOS GENÉTICOS (AGs)

Los Algoritmos Genéticos son algoritmos de optimización, búsqueda y aprendizaje que se basan en principios inspirados en los procesos de la evolución natural y de la genética.

La idea básica de estos algoritmos consiste en mantener una población de individuos que codifican soluciones del problema. Dichos individuos emplean una representación genética para codificar los valores de las características parciales que definen las distintas soluciones. Debido a ello, cada individuo recibe el nombre de cromosoma y cada una de sus componentes el de gen.

Los cromosomas se generan inicialmente a partir de la información previa disponible sobre el problema, o bien de un modo aleatorio cuando no la hay, y la población se hace evolucionar a lo largo del tiempo mediante un proceso de competición y alteración controlada que emula los procesos genéticos que tienen lugar en la naturaleza hasta que se verifique una determinada condición de parada como puede ser, por ejemplo, que se haya realizado un determinado número máximo de evaluaciones de individuos o que la población haya evolucionado un determinado número de generaciones.

A lo largo de sucesivas iteraciones, denominadas generaciones, los cromosomas se ordenan con respecto a su grado de adaptación al problema, es decir, con respecto a lo bien que resuelven dicho problema (usando una función que, devolviendo un valor numérico que se supone proporcional al grado de bondad de la solución que dicho cromosoma codifica, permite guiar al AG por el espacio de búsqueda) y, tomando como base estas evaluaciones, se construye una nueva población mediante un proceso de selección y una serie de operadores genéticos tales como el *cruce* (intercambia genes ya existentes entre individuos) y la *mutación* (permite alcanzar valores de genes que no estén presentes en la población, incrementando su diversidad genética), que se aplican con una probabilidad definida por el usuario.

En cuanto a su aplicación a los videojuegos, los algoritmos genéticos se pueden emplear para:

- Búsqueda de Caminos (PathFinding).

- Generar escenarios automáticamente para distintos niveles, y/o ajustar prioridades, comportamientos, u otros parámetros del juego.

- Estrategias orgánicas y enemigos adaptativos, consiguiendo una mejor inmersión mediante la evolución del comportamiento de los personajes: los parámetros (fuerza, intuición, valor, etc.) en base a los cuales se crea la personalidad o el comportamiento de un personaje pueden ser ajustados por el AG para que el desempeño de ese agente inteligente se optimice y adecúe mejor.

Requisitos del juego a desarrollar

Como comentamos al inicio, el objetivo de la práctica es programar un videojuego en **3D** utilizando **Unity** (http://unity3d.com/es), un **motor gráfico de videojuegos** multiplataforma empaquetado como una herramienta para crear juegos, aplicaciones interactivas, visualizaciones y animaciones en 2D y 3D creado por Unity Technologies que está disponible como plataforma de desarrollo para Windows, OS X y Linux.

El juego a desarrollar será de libre elección por parte del alumno, al igual que el lenguaje utilizado para programar los scripts (se puede usar indistintamente C#, JavaScript o Boo), aunque independientemente del juego elegido este debe cumplir los siguientes requisitos mínimos:

- **El juego a desarrollar ha de tener varias pantallas o escenas (Scene),** teniendo como mínimo una pantalla de presentación, otra de juego, otra de finalización y otra de configuración.
- **Los recursos (assets) utilizados en el juego** (sonidos, imágenes, texturas y materiales, música, etc.) **han de ser de libre distribución**, de forma que no se vulnere derechos de copyright (en la documentación habrá que indicar su procedencia y página de donde se han descargado, aun cuando sean de la propia Unity Asset Store).
- **La aplicación debe permitir guardar las preferencias más comunes del juego**, tales como volumen, records personales, partidas jugadas y cualquier cosa que necesitemos que se mantenga en memoria una vez se cierre nuestro juego.
- **La aplicación debe tener una pantalla de configuración** donde el jugador puede modificar aspectos del juego tales como número de vidas, tiempo para conseguir el objetivo del juego, etc.
- **Al finalizar el juego la aplicación debe mostrar la tabla de records personales**, permitiendo al jugador que inserte su nombre, en caso de que haya batido el record o quedado entre los 10 primeros.
- **Obligatoriamente, el juego debe hacer uso de** alguna de las nuevas técnicas de IA enumeradas en el apartado anterior (**Algoritmos Genéticos o Lógica Difusa**) **para modelar las decisiones y comportamientos que los personajes pueden tomar.**
- **Cuando esto no sea posible o no sea necesario** debido a las características del juego, **al menos deberá utilizar alguna de las técnicas de IA enumeradas en la práctica anterior** (Búsqueda de Caminos, Árboles de Decisión, Máquinas de Estado…) de forma que el comportamiento del juego sea inteligente.
- **Además del juego se deberá entregar una documentación** donde se explique las características del juego desarrollado, las clases y recursos utilizados para su elaboración y las direcciones web desde donde se han descargado dichos recursos, así como cualquier otro aspecto relevante que se desee considerar.

Evaluación

En la nota de la práctica se tendrá en cuenta tanto **el proyecto de programación** como el resultado de la entrevista y la defensa de las prácticas, **la calidad de la documentación entregada, la complejidad del juego desarrollado y la/s técnica/s de IA utilizada/s**, así como cualquier otro aspecto que deba ser tenido en cuenta como puede ser el número de integrantes del proyecto o el hecho de que los objetos del juego tales como música, sonidos, personajes, efectos especiales, etc. hayan sido creados por el propio alumno utilizando herramientas de modelado externas (aunque esto último no es necesario y no es objetivo de la asignatura se debe valorar el esfuerzo que el alumno realiza).

Tras la entrevista de prácticas, si la práctica está aprobada, se dará opción a mejorar la nota de la práctica corrigiendo los errores.

Las prácticas que no implementen TODA la funcionalidad requerida no serán evaluadas o tendrán una penalización en la nota final.

La copia de prácticas, aun siendo parcial, será sancionada con el suspenso de la práctica tanto para el grupo que copia como para el que deja copiar. Cada grupo es responsable de la custodia de sus prácticas.

Entrega de prácticas

La entrega de prácticas se realizará en moodle. El representante del grupo deberá subir la documentación de la práctica y el proyecto de programación en un fichero comprimido .zip o .rar (no olviden indicar los nombres de los integrantes del grupo).

La práctica será realizada en **grupos de dos alumnos** como máximo y será desarrollada en una entrega. Una vez finalizada la práctica se realizará la entrevista de revisión y la defensa de prácticas.

El plazo de entrega de la práctica y presentación en clase finaliza el lunes **30 de mayo** de 2022, disponiendo el alumno de una semana adicional para entregar y subir la documentación a moodle (hasta el lunes **6 de junio**).

El plazo para corregir la DOCUMENTACION, subsanar los errores y añadir las sugerencias propuestas tras la presentación de los videojuegos finaliza el **LUNES 13 DE JUNIO.**

Nota:
Moodle no admite ficheros de tamaño mayor de 50Mb, por lo que si el fichero a subir pesa más no será posible hacerlo. En ese caso podrán entregarlo a través del servicio de consigna de la uhu (https://consigna.uhu.es), de dropbox (https://www.dropbox.com/) o de cualquier otro repositorio, subiendo al moodle un .zip o .rar con la documentación y un fichero .txt con el enlace al repositorio donde se haya subido (si el enlace está protegido por contraseña no olviden indicarla).