Práctica 4: Dune

Fecha de entrenamientos: 20 y 27 de mayo de 2021
Fecha de entrega: 28 de mayo de 2021

Importante: Haz la entrega en tiempo y forma, subiendo al campus virtual un fichero *IAVP4GXX.txt* donde *XX* sea el número del grupo al que representas, con dos dígitos. Dentro de ese fichero incluye los datos del grupo, una breve explicación de la práctica, las ampliaciones realizadas o los problemas existentes y el enlace al repositorio donde estará el fichero *README.md* con toda la documentación técnica enlazada, la carpeta *IAVGXX* con todo el proyecto de la asignatura (plugins, recursos y todo el código fuente necesario para hacer funcionar vuestro controlador, que irá en una subcarpeta *RTSGXX* dentro de la carpeta *Scripts*), la versión ejecutable para Windows de 64bits *IAVP4GXX.exe* (publicada con sus carpetas y ficheros acompañantes) y el video comentado con vuestras pruebas *IAVP4GXX.mp4*.

1. Introducción

"Se habla de que han fortificado los poblados del graben *hasta tal punto que no conseguiréis nada contra ellos. Dicen que sólo necesitan sentarse tranquilamente tras sus defensas y dejar que vosotros os agotéis en fútiles ataques.*

—En otras palabras —dijo Paul—, se han inmovilizado.

—Mientras que vosotros podéis ir a donde os plazca —dijo Gurney.

—Es una táctica que he aprendido de ti —dijo Paul—. Han perdido la iniciativa, y esto quiere decir que han perdido la guerra.

Gurney sonrió con una sonrisa de complicidad.

—Nuestro enemigo se encuentra exactamente donde yo quiero que esté —dijo Paul. Miró a Gurney—. Bien, Gurney, ¿quieres enrolarte conmigo para el final de esta campaña?

—¿Enrolarme? —Gurney le miró sorprendido—. Mi Señor, nunca he dejado tu servicio. [...]"

Para este último desafío, el trasfondo será el universo de ciencia ficción que Frank Herbert inauguró en su exitosa novela *Dune*, publicada en 1965. La épica batalla final de los Fremen contra las fuerzas del Emperador y el Barón Harkonnen en el planeta desértico *Arrakis* (véase Figura 1) servirá para poner a prueba las técnicas de IA para Videojuegos sobre Evaluación y Coordinación que hemos estudiado. Diez mil años en el futuro, en nuestra misma galaxia, encontramos este planeta, también llamado *Dune*, que es el hábitat natural de los Gusanos de Arena, unas criaturas gigantes relacionadas con una escasa pero muy codiciada sustancia. Sólo

en este rincón del universo es posible extraer la especia conocida como *melange*, una droga que alarga la vida e incluso despierta poderes psíquicos en las personas que se ven afectadas por ella. Uno de los personajes principales de la historia es Paul Atreides, miembro de una estirpe nobiliaria caída en desgracia, que poco a poco se revelará como el *Kwisatz Haderach*, una especie de "mesías" que liderará la revolución del pueblo nativo de Arrakis, los Fremen, contra aquellos que explotan comercialmente los recursos de su planeta y son sus enemigos político-religiosos: el Imperio y sus socios de las Grandes Casas feudales que existen en la galaxia, como la liderada por el sádico Barón Vladimir Harkonnen.

Figura 1. Tropas y vehículos espaciales listos para el combate según la primera adaptación cinematográfica de la novela, también titulada *Dune* y dirigida por David Lynch en 1984.

El prototipo a desarrollar se construirá sobre un punto de partida que consiste en un pequeño juego de estrategia en tiempo real en el que pueden enfrentarse varios ejércitos, concretamente el bando Fremen contra las fuerzas Harkonnen. El *controlador* de cada ejército puede dar órdenes a las distintas unidades que posee para extraer recursos, moverse y atacar a sus enemigos. También puede usar sus instalaciones para crear nuevas unidades, si ha obtenido los recursos necesarios para ello. Este juego ya trae resuelta la percepción, el movimiento, la navegación y hasta las decisiones básicas de los agentes inteligentes que hay detrás de las distintas unidades, en forma de árboles de comportamiento. Por lo tanto, habrá que desarrollar un sistema de IA multicapa, en el que, a los agentes inteligentes, añadimos un controlador basado en IA que, con ayuda de un *mapa de influencia*, podrá jugar de forma autónoma tomando decisiones según la situación táctica del escenario de batalla que se encuentre a cada momento, intentando acabar con las unidades de su rival antes de verse destruido por ellas.

2. Planteamiento del proyecto

Desarrolla un prototipo de IA para Videojuegos, en forma de *controlador automático* de uno de los dos bandos enfrentados en el escenario de batalla del juego de estrategia en tiempo real proporcionado por el profesor (véase Figura 2). Este controlador deberá usar, al menos, un *mapa de influencia* (elaborado con los datos de ambos bandos) para tomar mejores decisiones sobre

qué órdenes dar a sus unidades e instalaciones en cada momento. Se pueden programar tantas clases como sean necesarias, aunque dicho código debe reunirse bajo una misma carpeta y espacio de nombres, y en todo momento hay que adaptarse a las interfaces y el funcionamiento de la infraestructura proporcionada por el profesor, sin poder modificar dicho "código base" ni acceder a él para modificar la situación de uno u otro bando de maneras que no sean las expresamente permitidas y contempladas en los comentarios de la API.

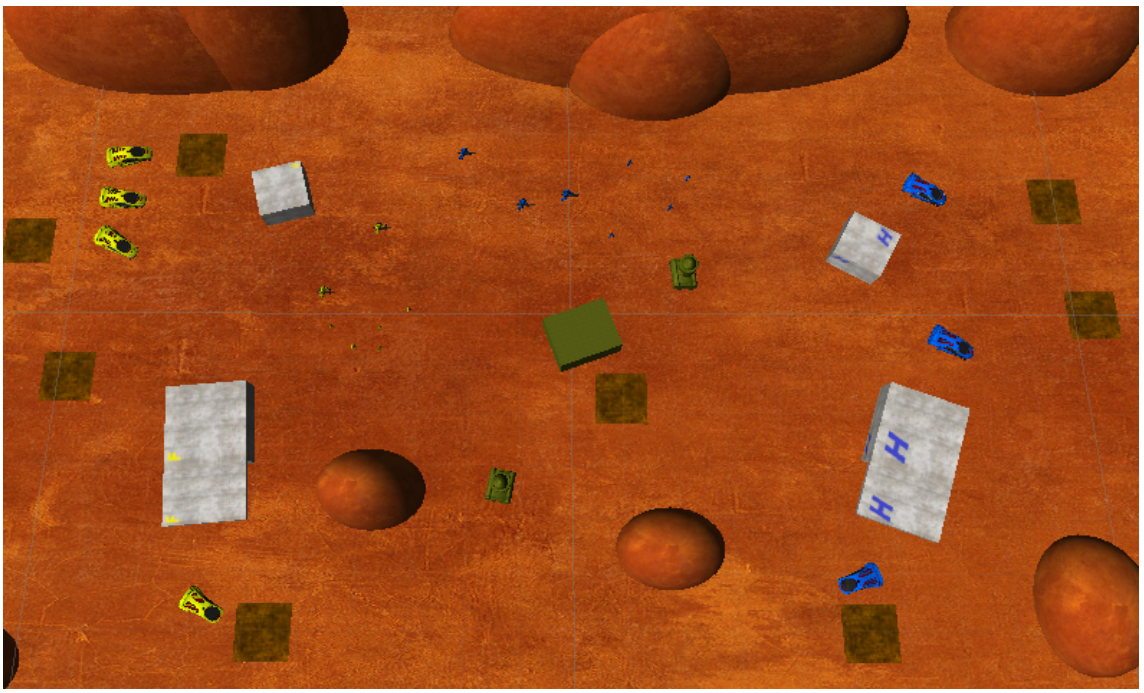

Figura 2. Captura del juego de estrategia en tiempo real utilizado como base. Las unidades extractoras, exploradoras y destructoras del pueblo Fremen son amarillas, mientras que las de la Casa Harkonnen son azules. Las instalaciones base y de procesamiento también llevan un estandarte del color correspondiente. Las torretas y poblados de los *graben* son verdes.

El escenario ya está preparado para su navegación, utilizando las mallas de navegación y se usa la percepción y el movimiento dinámico de Unity. Aunque debe conservarse el escenario de ejemplo para realizar allí pruebas de funcionamiento, el controlador debe estar totalmente desacoplado de cualquier escenario concreto y ser capaz de funcionar sobre otros **escenarios sustancialmente modificados**, siempre que cumplan una serie de condiciones razonables:

- Aunque pueda haber pendientes, **la navegación es posible** para todas las unidades entre cualquier par de puntos del escenario, no hay zonas inconexas.
- No hay **recursos** (campos de ***melange***) superpuestos ni en distribuciones rebuscadas. Para que la batalla sea justa se debe buscar cierta *simetría* en dicha distribución.
- Aunque haya **poblados *graben*** que no puedes atravesar, es posible destruirlos si reciben 10 puntos de daño. Las **torretas *graben*** además atacarán a cualquier unidad que se les acerque. Soportan 20 puntos de daño y causan 1 cada medio segundo (2 DPS).

- Podrá haber **obstáculos** sobre los que no se pueda navegar (típicamente ***dunas***), **estáticos e indestructibles**, a diferencia de las instalaciones, unidades, poblados y torretas, que no puedes atravesar pero sí pueden destruirse.
- Hay suficiente **espacio inicial** entre todas las instalaciones, torretas y poblados, e incluso entre las unidades que comienzan situadas en el escenario desde el principio y que, naturalmente, también deberían suponer un reparto *equitativo* de fuerzas.

En cuanto a las **instalaciones** y **unidades** de que disponen todos los controladores del juego, y al tipo de comportamientos que tienen, se ofrecen detalles a continuación:

- La **instalación base** es un edificio que hace las veces de barracón del ejército. Se le puede solicitar la creación de una nueva unidad de alguno de los tres tipos posibles: *extractora*, *exploradora* o *destructora*. Los requisitos para que dicha creación sea posible es que la instalación cuente con dinero (***solaris***) suficiente para ello y que no se supere el *límite* máximo de ese tipo de unidades. Puede recibir 100 puntos de daño antes de destruirse. Si se pierden las instalaciones base, se pierde la partida.
- La **instalación de procesamiento** es otro edificio que hace las veces de refinería del ejército. Se encarga de la conversión de la especia extraída en solaris, pero en realidad no se le puede solicitar hacer nada. Puede recibir 50 puntos de daño antes de destruirse.
- La **unidad extractora** es la responsable de extraer la especia melange de los campos donde esta se encuentra. Se le puede solicitar que se *mueva* a cierta posición, de modo que cuando se topa con estos campos, se pone a extraer. Tras realizar su trabajo, la unidad extractora irá a devolver su carga a la instalación de procesamiento, haciéndonos ganar los solaris correspondientes a los recursos extraídos. Salvo nueva orden, esta unidad repite el proceso *indefinidamente* e irá una y otra vez a ese mismo campo de especia a trabajar. En cada viaje se obtienen recursos por valor de 1.000 solaris. Puede recibir 10 puntos de daño antes de destruirse. Su fabricación cuesta 10.000 solaris y como máximo un controlador puede llegar a tener 5.
- La **unidad exploradora** es ágil para moverse por el escenario y también sabe combatir. Se le puede solicitar moverse a cierta posición y, o bien permanecerá allí inmóvil si la zona es tranquila, o atacará la instalación o unidad enemiga que encuentre cerca, así como la torreta o poblado *graben* que se interponga en su camino, hasta que dicho objetivo sea destruido. Tiene tendencia a perseguir a su objetivo si esta es una unidad enemiga que huye, y a contestar a su agresor, si en algún momento es atacado por una torreta o alguna otra unidad. Causa 1 punto de daño cada medio segundo (2 DPS) y puede recibir 5 puntos de daño antes de destruirse. Cuesta 15.000 solaris y como máximo se pueden tener 30.
- La **unidad destructora** es similar a la exploradora, más poderosa y resistente, pero también más lenta. Funciona de manera similar, aunque no persigue objetivos ni contesta a agresores, tendiendo más a centrarse únicamente en su objetivo a abatir. pero causa muchísimo más daño a los enemigos. Ataca causando 10 puntos de daño cada 2 segundos (5 DPS). Puede recibir 20 puntos de daño antes de destruirse. Cuesta 30.000 solaris y al mismo tiempo sólo se pueden tener 10.

El **controlador automático** representa al mando táctico supremo, una especie de “capitán general” de todo el ejército. Lo que tiene que hacer la IA que implementa dicho controlador es

dar órdenes a las unidades de su propio ejército, al igual que a su propia instalación base. Esto lo puede hacer en todos los *frames* o cada cierto periodo de tiempo (ej. utilizando corrutinas), pero teniendo en cuenta que las unidades necesitan tiempo para completar sus acciones y que las órdenes muy rápidas, si son contradictorias, pueden resultar contraproducentes.

Para poder *dar órdenes* contando con información táctica es necesario **sondear el entorno**, el estado del escenario y de las distintas instalaciones y unidades que allí operan, tanto propias como ajenas. La única técnica que debe ser usada obligatoriamente es la de la creación de un **mapa de influencia**, con un esquema de división de baldosas, por ejemplo, que permita conocer las zonas más o menos seguras, más o menos apropiadas para realizar según qué acciones. Para facilitar la depuración y la corrección, este mapa de influencia debe dibujarse de alguna manera sobre el escenario, reservando la **tecla M** para poder mostrarlo u ocultarlo.

La entrega será realizada en tiempo y forma **[1 pto.]**, el proyecto estará bien diseñado, organizado y comentado (importante agrupar todo el código del controlador en la carpeta *RTSGXX* con el espacio de nombres *es.ucm.fdi.iav.rts.gXX*) **[1 pto.]**, y la documentación explicará con claridad cuáles fueron las técnicas implementadas **[1 pto.]**, las pruebas realizadas y los resultados obtenidos (incluido todo escenario desarrollado para vuestras pruebas) **[1 pto.]**.

El prototipo ejecutable será usable y funcional, permitiendo:

- Mostrar en una primera ejecución, sin necesidad de tocar nada, el controlador automático **enfrentándose a sí mismo** en el ***escenario de ejemplo*** del juego de estrategia en tiempo real proporcionado por el profesor **[1 pto.]**.
- Incluir la posibilidad de probar *al menos* ***una variante del escenario*** desarrollada por vuestro grupo para probar el controlador automático, permitiendo enfrentar al controlador de IA **contra un jugador humano**, además de contra sí mismo (con los mismos o diferentes parámetros); siempre en *batallas 1 contra 1* **[1 pto.]**.
- Visualizar a voluntad el **mapa de influencia** utilizado, y cómo se va actualizando según cambia la situación táctica del juego **[1 pto.]**. Demostrar cierta **robustez en el funcionamiento** del controlador, tratando de evitar la *inactividad* o los *bloqueos* de las unidades, así como cierta *capacidad de reacción* a la hora de defenderse del enemigo y cierta *proactividad* a la hora de atacarlo para ganar la partida **[1 pto.]**.
- Demostrar mediante pruebas diseñadas a tal efecto un **nivel de competencia razonable en el juego**, aprovechando el tiempo y los recursos disponibles, venciendo sin problemas a *controladores incompetentes*, sean humanos o no **[1 pto.]**. Demostrar capacidad de adaptación tanto a diversas situaciones iniciales (empezar con o sin muchas unidades, con o sin mucho dinero, etc.) como a cambios bruscos en la situación táctica de la batalla (pérdida de las unidades, carencia total de solaris, etc.) **[1 pto.]**.

3. Restricciones y consejos

A la hora de desarrollar este proyecto es obligatorio:

- Restringirse exclusivamente a la programación del controlador automático, sin modificar ni usar de manera indebida el código ya proporcionado por el profesor. En caso de duda sobre la validez de algún planteamiento, se comentará con el profesor.

- Utilizar únicamente las herramientas de Unity y opcionalmente los plugins de terceros *Bolt* o *Behavior Designer*, sin reutilizar código ajeno al que proporciona el profesor.
- 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 implementados.
- Evitar, en la medida de lo posible, el uso de recursos audiovisuales pesados o ajenos.

Se pueden organizar todas las prácticas en un único repositorio, o incluso proyecto, siempre que se estructuren en distintas carpetas (especialmente *RTSGXX*), espacios de nombres, escenas, recursos, etc. Pensando tanto en las pruebas como en la revisión del profesor, y también con ánimo de reutilizar el esfuerzo de desarrollo entre ellas, conviene crear herramientas visuales cómodas para mostrar los mapas de influencia, escenarios de ejemplo intersantes y con instrucciones de uso, etc. El manejo debe ser ágil e intuitivo para poder repetir rápidamente todas las pruebas que sean necesarias con las variaciones que hagan falta.

4. Referencias y ampliaciones

Como punto de partida para la investigación, además de la bibliografía de la asignatura, puedes utilizar las siguientes referencias. En ningún caso debes replicar el código que encuentres por ahí; asegúrate de entenderlo y verifica que funciona *exactamente* como pide este enunciado.

- Bolt, Visual Scripting
  https://unity.com/es/products/unity-visual-scripting
- Opsive, Behavior Designer
  https://opsive.com/assets/behavior-designer/
- Unity, Navegación y Búsqueda de caminos
  https://docs.unity3d.com/es/2019.3/Manual/Navigation.html
- 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 allá en tu aprendizaje, puedes considerar estas posibles ampliaciones:

- Mejora aspectos del escenario, añadiendo sonidos o efectos visuales para mejorar su estética, manteniendo la estructura del código sin cambios.
- Mejora aspectos del movimiento y la navegación de las distintas unidades, aunque en ningún caso valdrá asumir que esas son las condiciones normales para las pruebas.
- Añade gusanos de arena, que se mueven lentamente por el nivel y tienen tendencia de ir hacia las instalaciones, causando daños al colisionar con otras entidades.
- Mejora el controlador para jugadores humanos, de manera que recibas *feedback* visual y sea posible precisar la posición a la que mandar moverse a las unidades seleccionadas.
- Localiza puntos de ruta tácticos para que las unidades naveguen mejor por el escenario.
- Realiza un análisis táctico de otro tipo distinto al de la creación de mapas de influencia.
- Coordina la acción de varias unidades de tu ejército, realizando jugadas predefinidas.
- Extiende la inteligencia del controlador para poder combatir en batallas *N contra N*.