Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Creación de un simulador y una IA sobre drones para la ayuda al rescate de montaña

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Cristian Pérez Bernal

Tutor: Javier Lluch Crespo

2017-2018

Resumen

Este proyecto se centra en la creación de un simulador que implementa un conjunto de algoritmos desarrollados en *scripts*, que ayuden a reducir el tiempo de búsqueda mejorando el rescate de una persona en un medio natural. Esta búsqueda se realizará con vehículos aéreos no tripulados, comúnmente llamados drones, los cuales se desarrollarán con unas características propias para la tarea encomendada, entre estas características tenemos un sistema de posicionamiento global y una cámara térmica, además de la forma y tamaño de cada uno de los drones. Entre estos algoritmos se pueden encontrar métodos para la creación de medios realistas en los que los drones desplegados tengan que enfrentarse a dificultades para ver su evolución. Durante la ejecución del simulador podrán encontrarse indicios de las personas desaparecidas, esto se gestiona desde una inteligencia artificial que funcionará mediante el uso de una máquina de estados la que le permitirá, mediante la detección de eventos, detectar tanto las colisiones con el terreno para evitar accidentes, como la detección de indicios que activarán subrutinas para la búsqueda exhaustiva por el terreno colindante y finalmente al encontrar el objetivo mandar su ubicación real haciendo las labores de rescate más simples y sencillas.

Palabras clave: inteligencia artificial, dron, Unity, 3D, terreno procedural, físicas, C#.

Resum

Este projecte se centra en la creació d'un simulador que implementa un conjunt d'algoritmes desenrotllats en scripts, que ajuden a reduir el temps de busca millorant el rescat d'una persona en un medi natural. Esta busca es realitzarà amb vehicles aeris no tripulats, comunament cridats drones, els quals es desenrotllaran amb unes característiques pròpies per a la tasca encomanada, entre estes característiques tenim un sistema de posicionament global i una cambra tèrmica, a més de la forma i grandària de cada un dels drones. Entre estos algoritmes es poden trobar mètodes per a la creació de mitjans realistes en què els drones desplegats hagen d'enfrontar-se a dificultats per a veure la seua evolució. Durant l'execució del simulador podran trobar-se indicis de les persones desaparegudes, açò es gestiona des d'una intel·ligència artificial que funcionarà per mitjà de l'ús d'una màquina d'estats la que li permetrà, per mitjà de la detecció d'esdeveniments, detectar tant les col·lisions amb el terreny per a evitar accidents, com la detecció d'indicis que activaran subrutines per a la busca

exhaustiva pel terreny limítrof i finalment al trobar l'objectiu manar la seua ubicació real fent les labors de rescat més simples i senzilles.

Paraules clau: intel·ligència artificial, dron, Unity, 3D, terreny procedural, físiques, C**#.**

Abstract

This project focuses on the creation of a simulator that implements a set of algorithms developed in scripts that help reduce search time by improving the rescue of a person in a natural environment. This search will be done with unmanned aerial vehicles, commonly called drones, which will be developed with their own characteristics for the task entrusted, among these features we have a global positioning system and a thermal camera, in addition to the shape and size of each of the drones. Among these algorithms can be found methods for the creation of realistic means in which the deployed drones have to face difficulties to see their evolution. During the execution of the simulator may be found indications of missing persons, this is managed from an artificial intelligence that will work through the use of a state machine which will allow, through the detection of events, detect both collisions with the ground to avoid accidents, such as the detection of signs that will activate subroutines for the exhaustive search for the surrounding terrain and finally when finding the objective to send their real location making the rescue tasks simpler and simpler.

Keywords: artificial intelligence, drone, Unity, 3D, procedural, physical terrain, C#.

Agradecimientos

A mi familia y amigos por el apoyo recibido

A Paulo por prestarse a realizar los vuelos de prueba.

A Ethan por ayudarme en el diseño y modelado del dron.

A Lucia por animarme siempre a mejorar el trabajo y apoyarme.

A Javier por toda la ayuda aportada y sus clases de Sistemas Gráficos Integrados.

Índice de contenidos

Índice de Imágenes

1. Introducción

Hoy en día las nuevas tecnologías constituyen un gran avance en muchos aspectos cotidianos de la vida de las personas, y la tendencia actual es la de abarcar cada vez más campos que la tecnología aún no abarca. Este trabajo quiere ofrecer la posibilidad de implementar nuevos métodos de ayuda en el ámbito del rescate de personas perdidas en situaciones en las que los métodos de búsqueda actuales son ineficientes.

Como se puede observar en la base de datos de personas desaparecidas y restos humanos (PDYRH) [1] gestionada por el gobierno de España en colaboración con todos los cuerpos policiales del país, desde antes del 2010 hasta la actualidad siguen activas casi 6.000 denuncias de personas desaparecidas y el número de las inactivas asciende a casi 140.000 personas.

A lo largo de este documento se expondrá la forma en la que actualmente se investigan las desapariciones y qué tecnologías se utilizan hoy en día, para después poder hacer una comparativa entre las diferentes opciones en materia de búsqueda y la herramienta desarrollada en este proyecto.

Cabe destacar que la motivación de este trabajo reside en ofrecer una mejora sustancial de los servicios públicos respecto a la búsqueda y el rescate de personas en el sistema montañoso nacional. Hay que tener en cuenta que España es un país que cuenta con un alto porcentaje de superficie como territorio montañoso y esto, junto al buen clima que goza durante todo el año produce una gran cantidad de alertas de personas perdidas. Este proyecto busca ayudar a mejorar la velocidad de resolución de estas desapariciones.

Por todo lo dicho anteriormente en este proyecto se desarrolla un simulador en el cual se puede observar el funcionamiento de un conjunto de drones, que trabajan de manera cooperativa para la búsqueda de un objetivo en común, encontrar a la persona o personas desaparecidas en un medio conocido como puede ser una montaña. Esto se realiza aplicando métodos de inteligencia artificial y técnicas para la simulación de terrenos realistas.

En este documento primero se habla de los objetivos que se plantean en este proyecto, después se expone como se diseña e implementa el proyecto, para finalmente acabar con los resultados mostrados. Posteriormente en la conclusión se recapacita sobre el cumplimiento de los objetivos, la metodología y herramientas utilizadas y finalmente los resultados.

2. Objetivos

En el presente proyecto se trabajan cuatro objetivos diferentes con el motor de videojuegos de *Unity3d*. Todos estos están relacionados con la creación de un entorno simulado y con la interacción entre los elementos que conforman el medio.

La misión del proyecto es ofrecer una herramienta al conjunto de investigadores y estudiantes del campo, para que puedan hacer simulaciones de situaciones reales, simplificando y facilitando la obtención de resultados sin la necesidad de hacer pruebas reales.

A continuación, se muestra una pequeña explicación de cada uno de estos objetivos para esclarecer su importancia en el proyecto.

Principalmente, se entiende por generación procedural aquella donde los contenidos no están diseñados de antemano y vienen almacenados con el propio simulador, sino que se crean en base a una serie de algoritmos definidos por los desarrolladores.

Dicho esto, y como primer objetivo está la creación del terreno procedural que dispone de una serie de algoritmos que permitirán la creación del terreno a partir de una imagen dada. También se le aplican texturas al terreno dependiendo de la inclinación de los polígonos que lo conforman para darle un aspecto más realista y finalmente se genera una capa que simula el agua, su movimiento y forma con una serie de algoritmos, aportando un realismo mucho mayor.

Al no tener gran cantidad de precedentes sobre el uso de drones en labores de búsqueda de personas desaparecidas se decidió diseñar y desarrollar drones propios, aplicándoles las mejores características de acuerdo con la discreción del autor a partir de la información recabada de diferentes fuentes.

Como segundo objetivo se desarrolla dos tipos de drones: un dron grande, a los que se llaman dron *master*, que está dotado de seis hélices y una cámara y otros más pequeños, a los que se les llama drones *slave,* los cuales solo tienen cuatro hélices y una cámara menos potente ya que son una copia más pequeña del *master.*

La parte más compleja a la hora de crear un simulador de drones y de la cual se plantea el tercer objetivo, es hacer que la navegación sea intuitiva y real, ya que no estamos manejando un dron con su propio control. Para mejorar este aspecto y hacer más fácil su manejo, se ha

optado por una visión en tercera persona con la cual se puede manejar el dron con el ratón de manera más precisa o con el teclado si se quiere que sea más veloz. Esta parte del simulador aporta un valor añadido ya que se puede observar el comportamiento de los drones en multitud de situaciones dando así una mejor experiencia de usuario.

Con el desarrollo de una inteligencia artificial se busca plantear el cuarto objetivo, que el conjunto de drones *slaves* y *master* se transformen en agentes racionales que perciban el entorno que los rodea y sepan reaccionar a diferentes situaciones para poder maximizar las posibilidades de éxito a la hora de buscar a una persona.

Para esto se desarrollan una serie de *scripts* y máquinas de estados cuyo objetivo principal es la búsqueda de las personas desaparecidas. Estas van a permitir al conjunto de los drones controlar las colisiones con los objetos que se puedan encontrar por el camino. Se ejecuta una formación aérea para mejorar el reconocimiento del terreno, se mantiene un nivel de vuelo adecuado para evitar la colisión con el terreno y finalmente marcará en el mapa la posición exacta donde se encuentra la persona.

3. Estado del arte

En este apartado se expresa el estado actual del mercado y las investigaciones respecto a las tecnologías actuales en materia de rescate y búsqueda de personas perdidas. También se abordarán algoritmos y métodos que se usan o están en fase de investigación para el mismo fin. Cabe destacar que es un campo, hasta ahora, muy poco desarrollado desde un punto de vista tecnológico ya que, por falta de fondos o medios no se ha visto popularizado.

3.1 Close-Search Project

El Close-Search Project [2] es un proyecto subvencionado por la Unión Europea (UE) y coordinado por el Instituto de geomática y cartografía de Cataluña en España desarrollado durante los años 2010 y 2012, cuyo objetivo principal es integrar en un helicóptero no tripulado, como el de la ilustración 1, una cámara térmica, un sistema de posicionamiento global (GPS) y una inteligencia artificial que le permita detectar a personas mediante las imágenes que capture.

Las pruebas de este proyecto se desarrollaron en Los Arcos, Navarra, en las que la plataforma voladora funcionó a una velocidad de 4m/s a una altura de 40 metros del suelo en un terreno de aproximadamente 500 metros por 300 metros, cabe destacar que el terreno elegido es una zona cuya máxima pendiente es de 3 metros ya que no presenta dificultades a la hora de controlar colisiones con elementos terrestres. Por otra parte, con la cámara térmica observaron que ciertos factores como la ropa y los objetos son despreciables ya que no aportan información alguna y tampoco ocasionan ruido a la hora de detectar a la persona.

Los resultados arrojaron un desfase entre la imagen y la posición del GPS de unos pocos metros y detectan a la persona en cuestión con una precisión aproximadamente del 95%, el coste total de este proyecto asciende a 450.000€ de los cuales subvencionó la UE hasta 300.000€.

Ilustración 1. Helicóptero utilizado en las pruebas del proyecto. Fuente: www.forodrones.com

3.2. Métodos de búsqueda

Hoy en día no se ha avanzado mucho en lo que a métodos de búsqueda de personas desaparecidas se refiere, por lo que se procede a explicar el actual método vigente para los cuerpos oficiales de seguridad recogido por la "Instrucción 1/2009, de la secretaria general de estado de seguridad sobre la actuación policial ante la desaparición de menores de esas y otras desapariciones de alto riesgo" [1].

Este protocolo indica que las desapariciones constituyen una de las principales preocupaciones a nivel social y también recoge unas conclusiones que nos permitan mejorar nuestra preparación ante estas situaciones, agilizar la capacidad de respuesta e incrementar la eficacia de las investigaciones.

El procedimiento que se sigue en el momento de la recepción de una denuncia, que es instanciada en la base de datos de PDYRH [1], es el siguiente:

- Traslado de la notificación a las patrullas de servicio pertinentes
- Determinar el grado de riego de la desaparición, la cual depende de múltiples factores indicados en el documento
- Activación de los métodos de búsqueda
- Localización de la persona desaparecida

Respecto al punto de "Activación de los métodos de búsqueda", para poder explicarlo con claridad se va a hacer referencia a la formación de los integrantes de Protección Civil [3]. En estos métodos hay que tener en cuenta aspectos como la información acerca de la víctima, información del suceso, factores medioambientales e información de la disponibilidad de los recursos. Se destaca también la gran tarea de organización que se lleva a cabo, la cual se basa en la jerarquización del conjunto de voluntarios e integrantes del cuerpo de Protección Civil.

En el método teórico la zona probable de búsqueda se traza mediante el uso de tablas, con las que se establece el área como una función de la distancia recorrida por el sujeto perdido, como se puede ver en la ilustración 2. Para esto se requiere determinar el último lugar donde la víctima fue vista por última vez (UPA). De este punto se generan dos superficies circulares de 1km y de 3km por donde se realizarán las búsquedas, siendo la primera más prioritaria que la segunda.

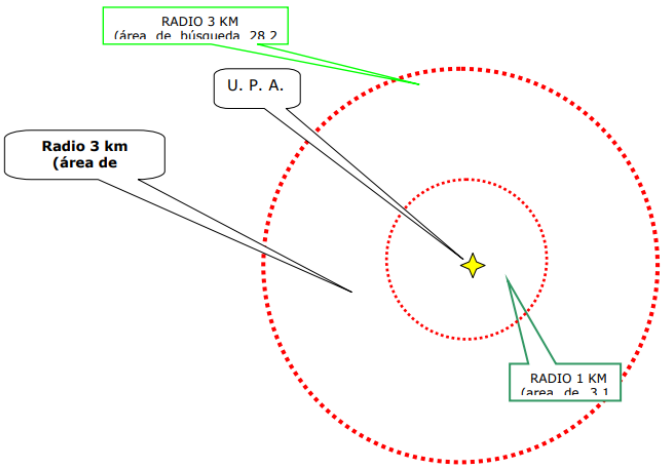


Ilustración 2. Método teórico [3].

El método estadístico se basa en el estudio del comportamiento de las personas extraviadas en la naturaleza, lo que proporciona los datos a este modelo. Se efectúan cálculos sobre las posibles distancias recorridas por los individuos, calculadas en línea recta. Este método genera una aproximación y siempre está sujeto a excepciones y las distancias calculadas pueden ser utilizadas para delimitar zonas con probabilidades de éxito.

Por otra parte, el método subjetivo, es un método usado para limitar áreas de búsqueda probable, se combinan un gran número de factores menos objetivos que con los métodos anteriores, como pueden ser los datos históricos, la intuición, la situación de accidentes geográficos o indicios y la consideración de las limitaciones físicas y psíquicas del sujeto. No obstante, este método ha sido de gran ayuda en numerosos casos, en especial en aquellos casos en los que la ausencia de un punto exacto del ultimo señalamiento dificultaba la situación de los datos teóricos.

El ultimo método presentado es el creado por el Teniente Coronel Rober Mattson, quien da nombre al método. Este procedimiento lo realizan dos o tres personas que analizan un mapa utilizando el método subjetivo para determinar el área en la que se efectuará la búsqueda. Esta técnica se basa en un proceso democrático en el cual todo el mundo, sin tener en cuenta el rango, la experiencia o el entrenamiento, participa de igual modo. Los cálculos utilizados son simples y no se requieren estudios sobre otros casos ni tablas de probabilidades.

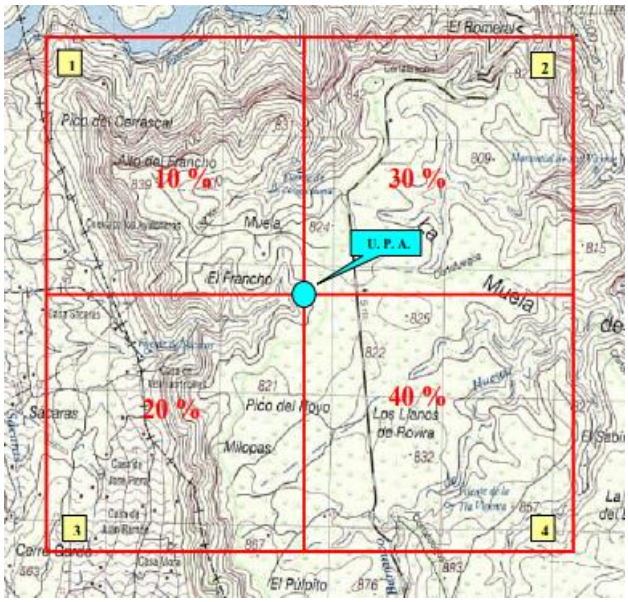


Ilustración 3. Método Mattson [3].

Las probabilidades distribuidas en la ilustración anterior se generan por la votación individual de cada uno de los participantes, estando estos obligados a que la suma de sus porcentajes sea cien por cien. Estas probabilidades están basadas en la intuición, la experiencia y la educación, eligiendo así las áreas más probables.

Después de esto se eligen las dos mayores probabilidades colindantes, siendo en el caso de la ilustración 3, las zonas 2 y 4 las que, sumadas, tienen mayor probabilidad que cualquier otra combinación y por eso se eligen como punto de inicio de la búsqueda.

En este método de segmentación del área de búsqueda es una técnica lógica. Esta búsqueda binaria se basa en la teoría de que suprimiendo zonas de un área de búsqueda donde no se hallen indicios, se reduce la extensión del área total y se pueden concentrar los recursos en los segmentos restantes.

Dado esto, los equipos de búsqueda deben prestar entonces atención a los indicios mientras atraviesan un área y concentrarse en la búsqueda de señales de la persona extraviada en los lugares donde existan mayores probabilidades de detectarlos.

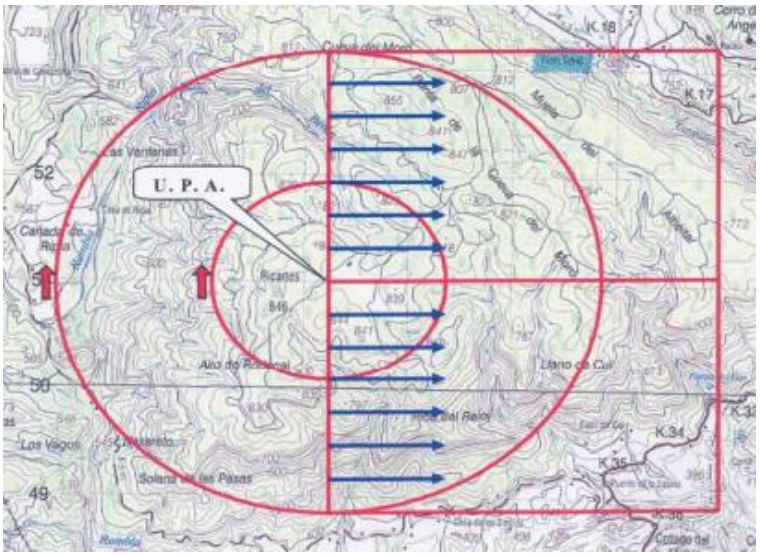


Ilustración 4. Orientación de la búsqueda una vez elegidas las zonas [3].

Para la mejor comprensión de la búsqueda binaria se va a proceder a explicar un caso hipotético, utilizado en la ilustración 5. Por ejemplo, un grupo de búsqueda "A" encuentra huellas en las subzonas 7 y 11 en dirección hacia la subzona 12 y el grupo "B" que recorre las zonas 13, 14 y 15 no encuentra nada. Por consiguiente, la casilla 11 es la que más probabilidades reúne y en la que se iniciará la siguiente batida.

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Ilustración 5. Zona dividida en subzonas [3].

Como conclusión y basándonos en los métodos ya expuestos, los grupos mínimos de buscadores que se necesitan serían ocho, compuestos de al menos diez personas cada uno, para rastrear el mismo terreno. Aunque el método Mattson reduce drásticamente estos números, siguen siendo excesivamente altos para la velocidad con la que es ejecutado y teniendo en cuenta que se basan en indicios y no exploran el terreno en su totalidad. La propuesta de este trabajo busca mejorar el tiempo de búsqueda y reducir el material humano a unas pocas personas, en el que muchas búsquedas son insuficientes.

3.3. Drones

En este apartado se comienza definiendo el concepto de dron, los tipos que existen, las partes de las que está formado y las herramientas que existen para poder manejarlos. La información aquí mostrada sirve para entender con mayor facilidad el resto del documento.

Un dron es «una aeronave que vuela sin tripulación, reutilizable, capaz de mantener de manera autónoma un nivel de vuelo controlado y sostenido, y propulsado por uno o varios motores» [4]. Lo que hace interesante y llamativo a estas aeronaves es la capacidad de operar de forma autónoma.

3.3.1 Tipos de drones

Las clasificaciones de drones son bastante variadas en la actualidad, pero en esta sección se van a exponer los drones clasificados en función de su misión, el tipo de alas y la manera de controlarlos. Según la misión, se pueden considerar las siguientes clases:

- *Blanco*: drones de uso militar que sirve de blanco para pruebas de artillería o bélicas.
- *Reconocimiento*: este tipo de drones son usados para reconocer terrenos y posibles objetivos.
- *Combate*: drones usados para atacar zonas de conflicto sin poner en riesgo vidas.
- *Logística*: drones especializados en la carga de objetos, Hoy en día se pueden ver desarrollos de drones transportistas.
- *Investigación*: estos drones se usan para hacer pruebas de nuevas tecnologías que *posteriormente* se implementarán en drones útiles.

La siguiente clasificación se basa en la forma del dron, especialmente al tipo de alas que usa [5].

- *Ala fija*: se componen de un ala rígida, con una determinada superficie de sustentación que, asociada a una hélice normalmente localizada en la parte posterior, le permite auto-propulsarse gracias a un motor eléctrico o de combustión.
- *Multirrotor*: dron formado por varias hélices que giran alrededor de un mástil. Existen múltiples configuraciones para este tipo, pero los usados en este proyecto son los cuadricópteros y los hexacópteros, que tienen cuatro y seis hélices respectivamente.

La última clasificación de la que se va a hablar es la que distingue la forma en la que los drones son controlados, según el método utilizados le permite más o menos autonomía.

- *Autónomo*: dron que utiliza sus propios sistemas y sensores, integrados en el mismo dron, que se usan para guiarse y evitar posibles colisiones con obstáculos que puedan aparecer por el camino. De esta forma no es necesario un piloto.
- *Monitorizado*: el dron sigue su propio plan de vuelo, pero necesita supervisión humana. Se le proporciona la información necesaria para el vuelo.
- *Supervisado*: el dron realiza algunas tareas automáticamente, aunque es pilotado por un humano.
- *Controlado remotamente*: Son controlados mediante un piloto con una emisora.

3.3.2 Partes de un dron

Una vez expuestas las distintas maneras en las que se pueden clasificar, se explica que partes lo componen y para qué sirve de cada una. Esta explicación se centra en los drones multirrotor, ya que es el tipo de dron en el que está centrado este trabajo.

- *Frame*: el frame o esqueleto principal de un dron, es donde van instalados el resto de los componentes. Pueden ser de varios tipos de material y formas dependiendo del uso.

Ilustración 6. Frame de un dron. Fuente: www. dronematters.com

- Motores: los motores son los que se encargan de proporcionar la velocidad de giro a las hélices, lo que genera una fuerza de empuje hacia arriba y permite el vuelo del dron. Muchas veces estos van recubiertos por un protector para evitar las colisiones con elementos que no han sido detectados por los sensores de proximidad, como ramas o pájaros.

Ilustración 7. Hélice montada en un motor. Fuente: www.surehobby.com

- Batería: en los drones pequeños suele usarse una batería de polímero de litio como fuente de energía, mientras que en los drones más grandes se usan motores de combustión con depósitos de combustible.

Ilustración 8. Batería de 6000 mAh. Fuente: www.ardupilot.org

- Modulo GPS: este módulo sirve para proporcionar al controlador del dron su ubicación geo-referenciada. Esta parte es importante ya que le brinda al dron el poder seguir una ruta predeterminada y le da al piloto la posibilidad de saber la posición del dron en cualquier momento.
- Telemetría: este componente le permite al dron establecer comunicación con su controlador con una serie de protocolos propios. También se comparten datos sobre la velocidad de vuelo, la posición, la dirección, etc.

4. Herramientas utilizadas

A continuación, se va a hacer una breve explicación de las herramientas utilizadas para desarrollar este proyecto, indicando los aspectos más utilizados y sus principales características.

- *Unity3D*: es el entorno donde principalmente se ha desarrollado el simulador, creado por *Unity Technologies* en el año 2005 y mejorado hasta la actualidad. Ahora se detallarán los elementos más relevantes del editor.
    - *Navigation*: esta herramienta interna del editor nos permite controlar las partes del escenario actual que son transitables para el elemento *player*, que en este caso es el conjunto de los drones. Se genera así una subcapa azulada encima del terreno que te indica por donde puede moverse.

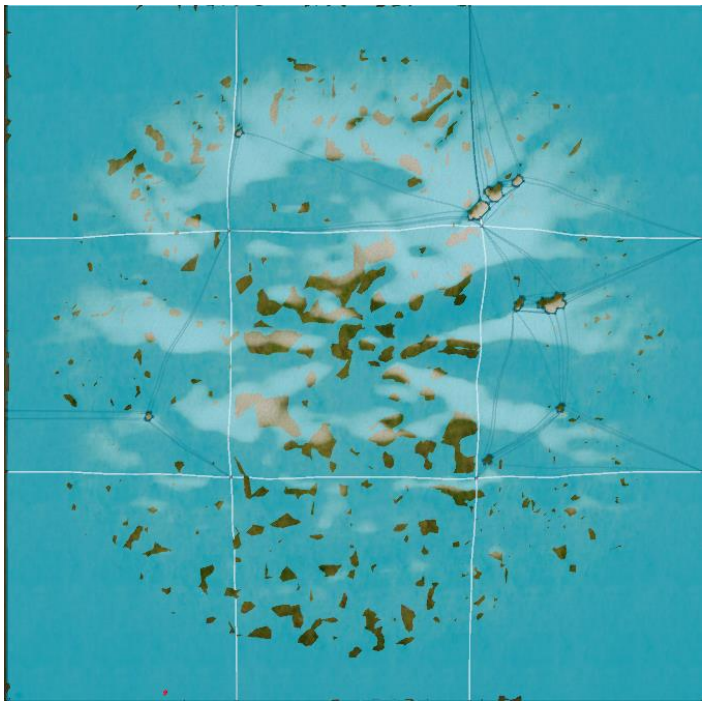


Ilustración 9. Las zonas azules del mapa muestran los lugares transitables por el dron.

    - *Animation*: para generar y controlar animaciones básicas se tiene esta herramienta que permite crear varias animaciones para cada objeto y modificarlas, dando así múltiples posibilidades a la hora de animar un objeto en tiempo real.

    - *Editor*: todo el entorno se basa en esta ventana principal en la que tiene lugar todo el desarrollo a excepción del desarrollo de *scripts*. Da una gran variedad de posibilidades y junto a su fácil uso hace que el desarrollo en él sea ligero y fluido.
- *Visual Studio*: en este proyecto se ha optado por el uso de esta herramienta en vez del propio desarrollador de *scripts* que tiene *Unity.* Permite una depuración mucho más severa y está plenamente integrado con el lenguaje elegido para el desarrollo del simulador, que en este caso es *C#*.
- NX o Unigraphics: paquete software de diseño asistido por ordenador (CAD), desarrollado por Siemens, que permite un diseño a un nivel avanzado y con el que se modela los drones.
- Pix4D: es un conjunto de productos de software que utiliza algoritmos de fotogrametría y visión artificial para transformar un conjunto de imágenes georreferenciadas en un modelo o malla de tres dimensiones (3D).

5. Desarrollo del simulador

Para que quede constancia del proceso de creación del simulador y los pasos seguidos para ello, se expone como se idea el proyecto partiendo desde el diseño hasta la implementación, y cada una de sus funcionalidades. En cada uno de estos apartados se comentan los puntos más relevantes, así como la construcción de sus *scripts*, objetos del simulador y componentes que lo hacen funcionar, de forma que al final de este apartado se tendrá una idea completa del funcionamiento del simulador.

5.1 Diseño

La calidad del trabajo final recae en un buena organización y diseño que a posteriori permiten al desarrollador adecuar nuevos elementos al programa principal, sin la necesidad de hacer cambios significantes en la estructura de este. El programa está diseñado en base a dos grandes bloques que dentro contienen más elementos del juego para generar una jerarquía que le otorgue al desarrollador la libertad necesaria para añadir nuevas funcionalidades.

Se definen dos tipos de niveles principales alrededor de los cuales se implementarán todas las funcionalidades. Estos niveles son:

- *Canvas*: este contenedor tiene todas las pantallas y elementos visuales del *Head-up display* (HUD) o también “barra de estado” que van a servir para darle indicaciones al usuario o mostrarle cualquier información.
- Controlador: el cual es el encargado de contener todos los objetos de escena y contiene los siguientes elementos:
    - *Directional Light*: este elemento genera la luz dentro de la escena, se ubica en una posición fija y es configurada para que genere la máxima resolución posible.
    - *Camera*: de este elemento se obtiene la visión del espectador o usuario y contiene un elemento que sirve para post-procesar y que mejora varios aspectos de la imagen que se explicaran más en adelante.
    - Terreno: como bien indica su nombre esta parte va a contener el terreno y todos los *scripts* que hagan falta, pero también tendrá varios elementos del simulador como el agua, el objetivo o persona que se busca y un objeto a modo de pista que se modelará como una bandera.

- *DroneController*: se encarga tanto de contener los objetos que simulan a los drones como de manejarlos, así como las animaciones y todo lo que sea necesario para el buen funcionamiento de estos.

El sistema que se ha expuesto es importante tenerlo presente desde el principio ya que se van a hacer múltiples menciones a él en los apartados siguientes.

5.1.1 El terreno

Ofrecer un simulador sin un terreno correctamente diseñado y con unos rasgos poco realistas sería inútil, ya que a la hora de comprobar los resultados arrojados perderían toda la veracidad, por lo que se propone un terreno creado a partir de una imagen en escala de grises de un terreno ya conocido, como el de la ilustración 10.

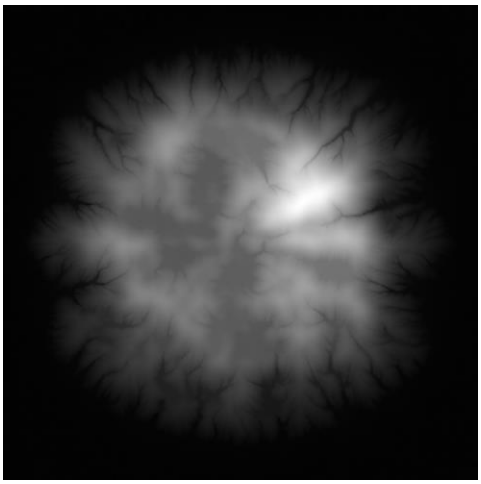


Ilustración 10. Imagen de un terreno simulando una montaña.

Para poder crear el terreno vamos a crear el objeto con las mismas dimensiones que la imagen. Por cada pixel de tamaño de la imagen habrá un punto en la malla del objeto y para definir las diferentes alturas se va a obtener el valor del brillo de cada uno de los pixeles de la imagen, aplicando ese valor al vector de altura al punto que corresponda en la malla.

La solución que se espera es un terreno con la forma y altura que nos indica la imagen, aunque sin color alguno ya que el proceso previamente comentado no obtiene esa información de ninguna fuente. Se desarrolla un algorimo que clasifique la inclinación entre todos los puntos del mapa, otorgandoles un valor entre cero y uno, para para poder aplicarle una textura diferente en cada rango que definamos como se ve en la ilustracion 11.

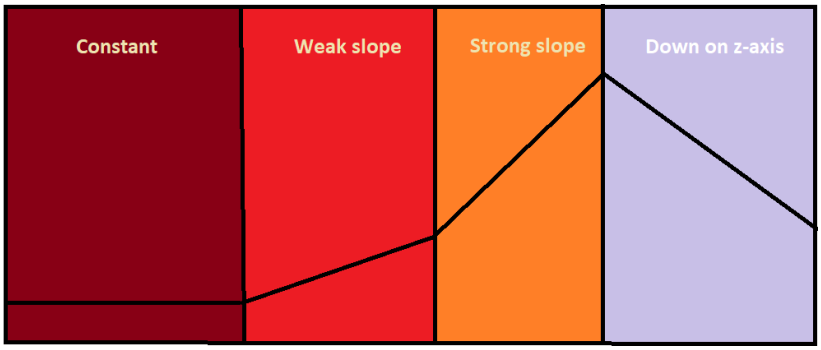


Ilustración 11. Explicación de los cuatro umbrales que contempla el algoritmo.

Ahora se procede a explicar los diferentes componentes que conforman el terreno y no forman parte integra de él, es decir son objetos diferentes que dependen del terreno y que tienen una importancia notable en la ejecución del simulador.

Primero tenemos los indicios o pistas para el dron, ya que mejoran la búsqueda del objetivo final, indican al conjunto de drones que la zona en la que está trabajando está cerca de la persona en cuestión. Estos elementos se representan con la forma de una bandera con una textura llamativa para hacer obvio su cometido.

Estas banderas se disponen de manera semi-aleatoria cerca del objetivo, pudiendo generarse automáticamente en cualquiera de los cuatro puntos cardinales de la anterior bandera creada y obteniendo la altura del elemento del terreno. Se dice de manera semi-aleatoria porque al generarse sobre unos puntos ya establecidos no hay muchas opciones. También es semi-aleatoria la distancia a la que se instancian desde la anterior bandera ya que se acota un rango de metros mínimos y máximos, entre diez y quince metros.

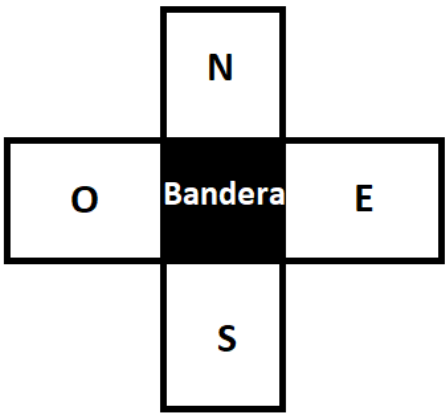


Ilustración 12. La generación de la siguiente bandera se hace en uno de los puntos cardinales indicados en la imagen.

Después tenemos el elemento del simulador que usaremos de objetivo para el dron. Este elemento se genera automáticamente en el terreno y a la altura correspondiente para que no aparezca en lugares inaccesibles, y se representa con una forma simple y de color llamativo para que resalte sobre el resto.

Y por último se explica el elemento que simula al agua. Este elemento se crea a partir de una malla prefabricada a la que se le aplica una serie de movimientos por cada *frame* del simulador para que, dé la sensación de movimiento y se le aplican ciertas texturas para que parezca más realista, haciéndola más agradable la vista.

5.1.2 Los drones

Como ya se ha comentado anteriormente, el dron se ha diseñado en base a la información recabada de diferentes fuentes. Esta información permite obtener valores desconocidos hasta ahora como el número de aspas por hélice, la longitud de los brazos o el mejor material del que puede estar fabricado dependiendo de las situaciones a las que se va a ver inmerso.

Base: el dron se basa en un armazón redondo con una altura de 22cm y una longitud máxima de 96cm para el dron *master* donde tiene seis brazos dispuestos cada 60 grados alrededor del centro de la base y una altura de 7,2cm y una longitud máxima de 32cm para los drones *slave*, por la parte de abajo en ambos drones se acopla una cámara y dos patas trapezoidales que le permitirán aterrizar.

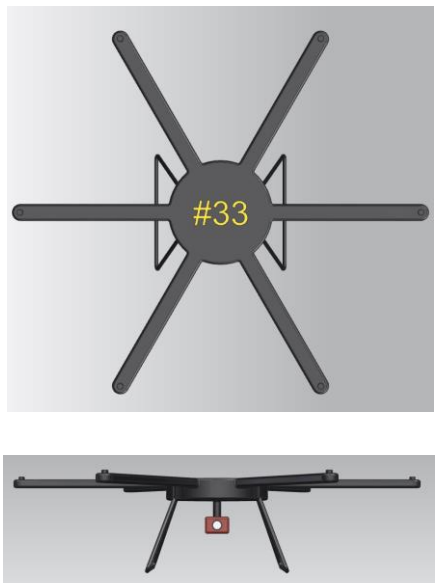


Ilustración 13. Base del dron donde se puede ver las patas trapezoidales y la camara .

Hélices: por cada brazo del dron se dispone de una hélice compuesta por 3 aspas de 12cm de longitud para el dron *master* y de 4cm para el dron *slave*. Las hélices son del mismo color excepto dos de ellas que tienen un color diferente para remarcar la dirección a la que está orientado.

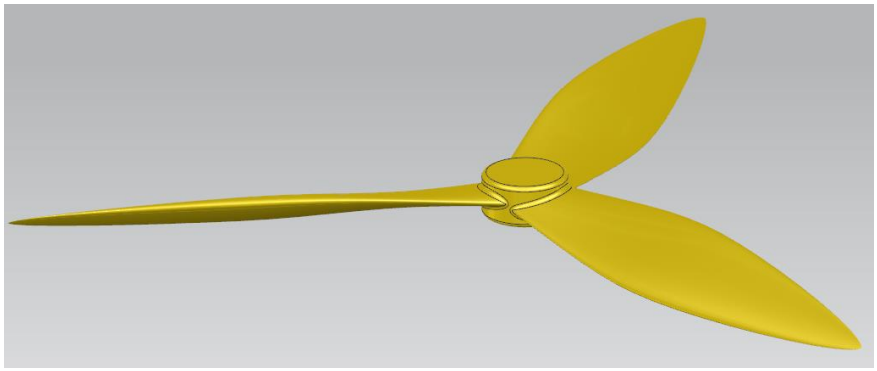


Ilustración 14. Aspas guía de color amarillo y forma aerodinámica del aspa.

Los drones pequeños que acompañan al *master*, modelados a partir del mismo, tienen dimensiones menores para mejorar el manejo y reducir el consumo de energía.

Una de las cosas a tener en cuenta con el diseño de los drones son las partes móviles de este, en concreto las hélices. Estas deben de poder rotar libremente, aunque ya estén ensambladas en el armazón.

Por otra parte, también se tiene que tener en cuenta la dirección de la rotación de las hélices para una correcta sustentación de la aeronave, por eso la colocación de las hélices tiene que ser diagonalmente opuesta para los cuadricópteros, de tal modo que si miramos el dron desde arriba la esquina superior derecha y la esquina inferior derecha rotaran en la misma dirección. En él caso de los hexacópteros si numeramos las hélices en sentido horario o antihorario, las hélices pares rotarían en una dirección y las impares al contrario o viceversa.

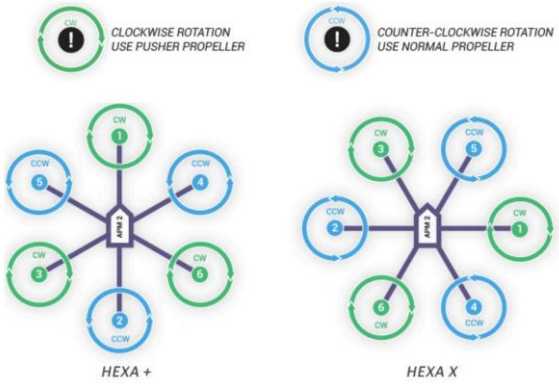


Ilustración 15. Dirección de la rotación de las hélices Fuente: forum.erlerobotics.com.

5.1.3 La inteligencia artificial

Para que la idea tome forma y dé unos resultados aceptables, es indispensable una buena inteligencia artificial que gestione toda la información que se capta del entorno y tenga los recursos necesarios para actuar en cualquier situación.

Para que el sistema actúe de manera reactiva, se crea una máquina de estados implícita en el código que consta de una serie de funciones que se ejecutan tanto al activar un estado como al salir de él. Hay que indicar que todos los estados del programa tienen su propio sistema de eventos y una función propia que se actualiza con cada *frame* del simulador.

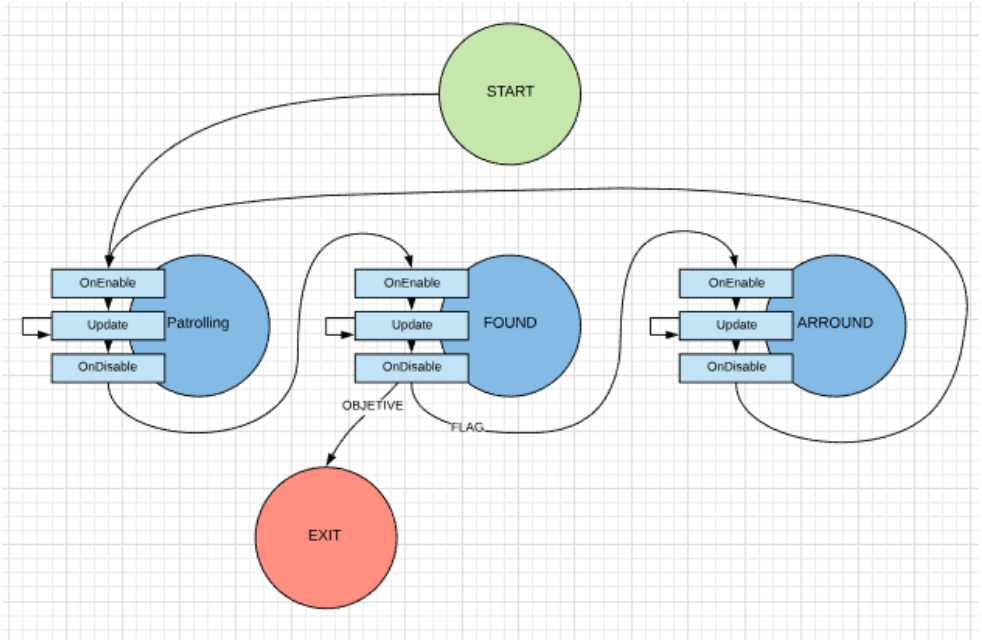


Ilustración 16. Ejemplo de máquina de estados, funcionamiento de la búsqueda del objetivo.

En la ilustración 16 podemos ver el funcionamiento de la máquina de estados, que se encarga de gestionar el comportamiento del conjunto de drones. En el comienzo del programa o *start* los drones aparecen en la escena y automáticamente pasan al estado de búsqueda o *patrolling* el cual determina el siguiente punto al que tienen que ir. Si al llegar no han encontrado nada se genera otro punto, y así consecutivamente hasta recorrer por completo el terreno que se le ha proporcionado. En el caso de que encuentre algún tipo de elemento, determina si es un indicio o el objetivo. Si lo que encuentra es el objetivo, envía la señal de socorro y se mantiene en el lugar, pero en el caso de que sea un indicio activa el estado de búsqueda cercana o *arround* en la que los drones esclavos se extienden por el terreno aproximadamente cinco metros en línea recta para buscar más indicios. Si no se encuentra el objetivo o un nuevo indicio, se reinicia el estado de *patriolling.*

5.1.4 La interfaz

Una de las partes más importantes de cualquier programa para los usuarios, es la interfaz, ya que le da al usuario el poder de manejar la aplicación a su gusto, otorgándole un alto grado de libertad sobre la configuración de las opciones del programa. Por eso se desarrolla un sistema de ventanas superpuestas a la visión principal del usuario como se ve en la ilustración 17, las cuales se van a describir más detalladamente a continuación:

- Ventana de inicio: esta ventana contiene varios elementos de la configuración inicial como botones palanca o *toggle button* que permiten al usuario activar o desactivar ciertos aspectos del simulador, así como un botón de inicio que pondrá en marcha todo el simulador tomando las características de los botones anteriormente nombrados.
- Ventana de ayuda: este elemento contiene una leyenda del uso de cada tecla y su función. Se puede ver durante toda la ejecución del simulador apretando una tecla concreta que se notifica al inicio del simulador para que el usuario lo tenga en cuenta.
- Ventana final: en esta ventana el usuario puede ver el tiempo total, en segundos, que ha tardado el conjunto de drones en cumplir su objetivo. También tiene varias opciones en las cuales se encuentran el reiniciar la simulación y salir del programa.

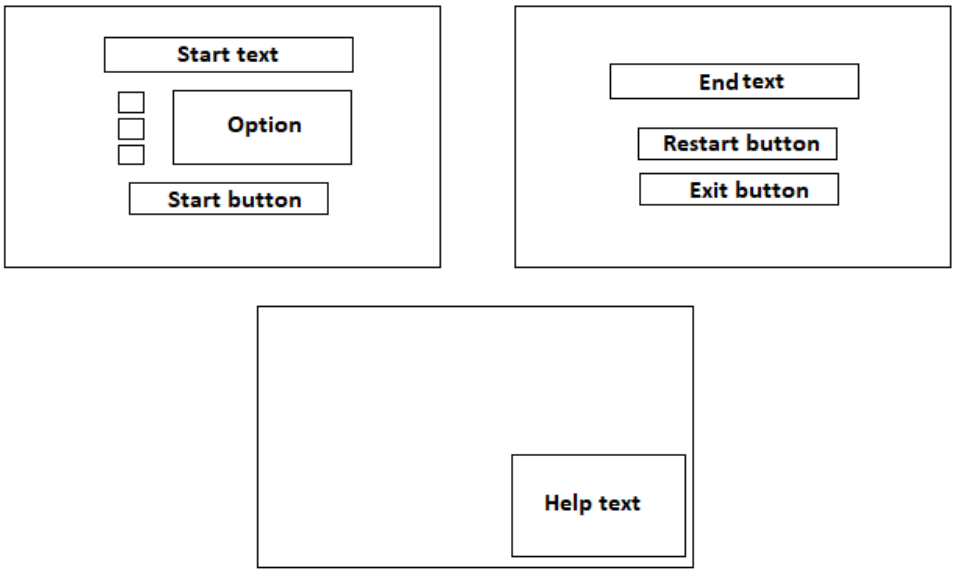


Ilustración 17. Boceto de las ventanas a desarrollar.

5.2 Implementación

Para que todas las ideas expuestas en este documento tomen forma y sirvan para un fin, se precede a explicar cómo se han implementado cada una de las partes explicadas en el apartado de diseño. Se utiliza el Editor de *Unity* y el entorno de desarrollo de *Visual Studio* de Microsoft para generar toda la arquitectura que se necesita para el desarrollo del simulador. Como ya se dijo anteriormente, el lenguaje de programación utilizado es *C#*.

En esta sección del documento profundiza más en la forma en la que se ha desarrollado cada uno de los *scripts* que conforman este proyecto, de los cuales se explica las partes que los componen, cómo funcionan y su coste computacional.

5.2.1 El terreno

Lo primero que se nos ocurre a la hora de diseñar un terreno, es hacerlo con la herramienta propia de *Unity* que permite hacer desniveles en el terreno con un simple clic, y aplicar texturas arrastrándolas directamente desde el directorio o usar un *asset* de la tienda para que lo genere automáticamente. Puesto que la idea a desarrollar es hacerlo en base a una imagen en escala de grises, no se pueden utilizar estas técnicas porque no permiten el uso de imágenes. Sabiendo esto se genera un *script* propio el cual ahora se explica.

Este *script* consta de dos funciones:

La función *LoadPNG* recibe como parámetro una cadena de caracteres, la cual se reconoce como la ruta dónde está guardada la imagen a analizar. Posteriormente se obtiene de esa imagen una matriz de *bytes*, la cual se carga en una textura de *Unity* mediante la función *LoadImage* que nos proporciona la clase *Texture*. Así conseguimos tener la imagen dentro de un objeto que el entorno conoce y poder obtener de ahí los datos.

La función *Awake* es propia de la interfaz de programación de aplicaciones (API) que nos proporciona el entorno, su función es ejecutar el código que haya dentro de la función antes de que comience el juego. Esta función se divide en dos grandes bloques.

El primero se encarga de usar la función *LoadPNG* para obtener la textura de la imagen. Hecho esto obtenemos los metadatos del terreno para comparar los tamaños, esto se hace porque, si cualquiera de los tamaños difiriese, el objeto final se vería deformado. Una vez tenemos los tamaños iguales procedemos a normalizar las posiciones de la imagen con el tamaño real y obtenemos, a partir de la función *Mathf.Floor*, el valor más largo posible para minimizar el

error de transformación. Todo esto se hace tanto para todos los vectores X de cada vector Z, aunque en la imagen el vector real utilizado es el Y ya que es 2D y al hacer la transformación en 3D se queda en el vector Z. Los cálculos se hacen a pares para después poder interpolar el punto anterior con su siguiente, generando así la malla que después aplicaremos al terreno dándole el aspecto que deseamos.

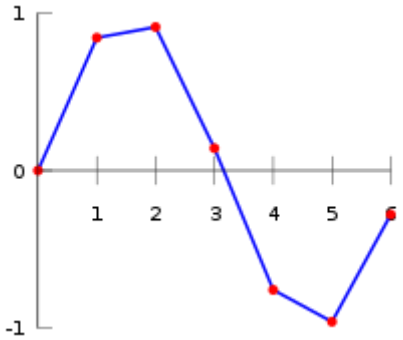


Ilustración 18. Ejemplo de interpolación. En este caso, primero se genera los puntos rojos de dos en dos y finalmente los uniría mediante la interpolación de estos. El eje vertical corresponde al vector Y y el eje horizontal pertenece al vector X.

Posteriormente, una vez tenemos ya la nueva matriz de alturas completa, asociamos cada punto de esta matriz a su altura obteniéndola del valor *greyscale* o escala de grises de la imagen original. Después de esto usamos la función *SetHeights* que nos proporciona la clase *TerrainData*, clase que deriva de la clase *Terrain* y que contiene todos los metadatos de ese objeto, para asociar la malla al objeto del terreno.

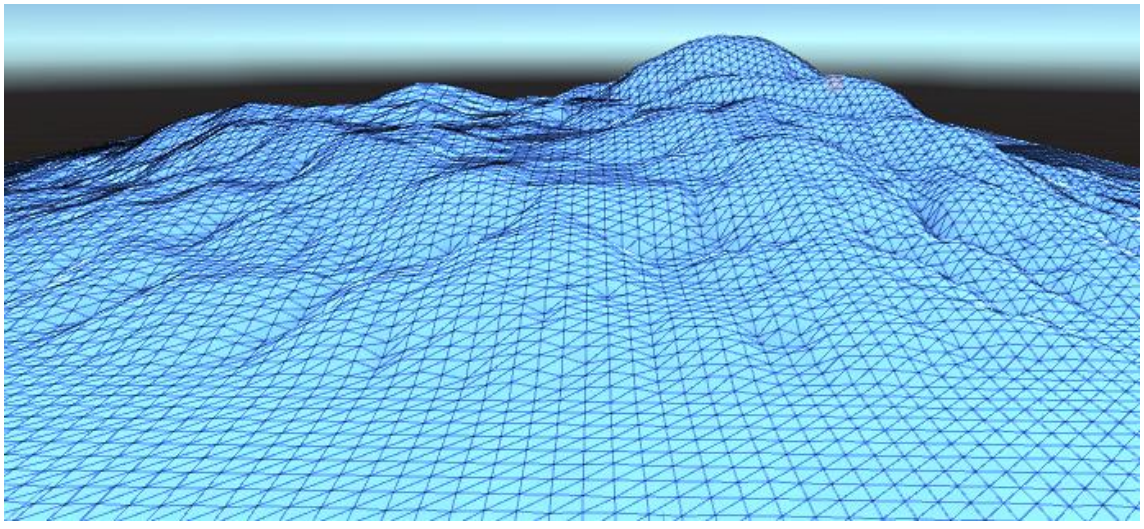


Ilustración 19. Resultado de aplicar la matriz de alturas a la malla del terreno.

El segundo bloque se encarga de darle un aspecto realista y se encarga de aplicarle una serie de texturas a la malla del terreno y a crear una capa de agua para la simulación del mar.

Para aplicar las texturas, lo primero que se hace es añadir las que parecen más adecuadas para este tipo de terreno en el editor de *Unity*. Para ello se hace clic en el objeto del terreno y se accede a la sección de *Paint texture* donde se permite asociar a ese terreno las texturas que tengamos en la raíz del proyecto o en los *assets*.

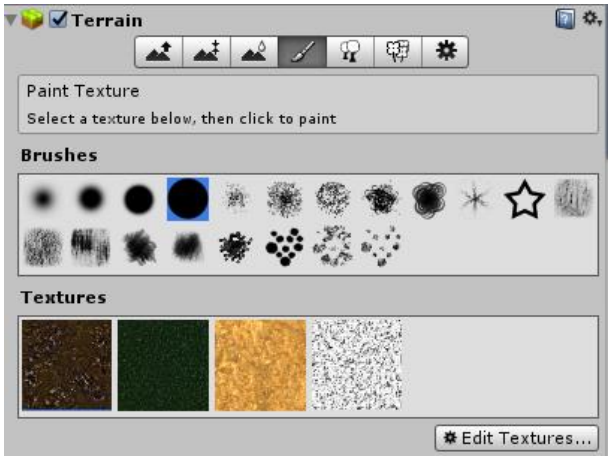


Ilustración 20. Ventana donde añadir las texturas en el editor de Unity.

Ahora desde el *script* asociado se obtiene de cada punto de las coordenadas X e Y de todo el terreno su normal para poder obtener la inclinación de ese punto, dependiendo del valor de esa inclinación se le aplicará una textura u otra. Cabe resaltar que todas las texturas tienen que ser acumulables, es decir que cuando una inclinación sea muy pronunciada se le aplique la última textura añadida y las anteriores. Los umbrales son los indicados en la ilustración 8: el primer umbral aplicará una textura constante a todo el objeto, el segundo se manifiesta de manera más pronunciada en bajas altitudes, la tercera textura se asocia al umbral que determina si un terreno es plano y la última textura depende de la altura, siendo más fuerte cuanto más alto, pero solo en las pendientes hacia el eje Z.

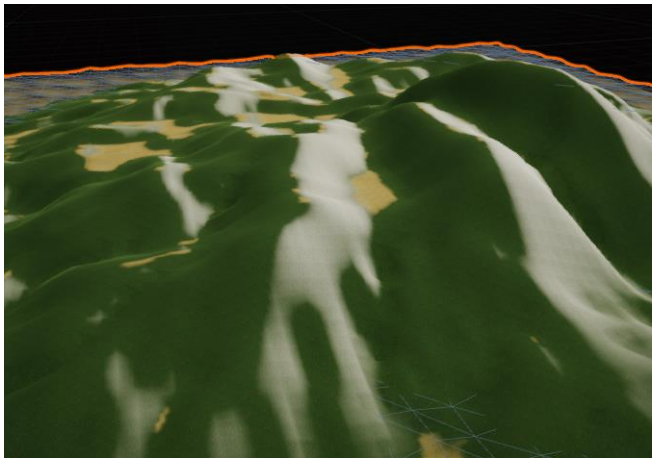


Ilustración 21. Muestra de la aplicación del algoritmo de coloreado.

Para finalizar se obtiene la capa del agua, previamente creada en el editor. Se activa usando la función *SetActive*, que permite activar o desactivar objetos del juego. Ademas, usa la posición del terreno para centrar la capa del agua, así como su tamaño, usado en los puntos anteriores, para darle las mismas dimensiones.

5.2.2 Los drones

Esta sección se compone de varios *scripts* que controlan todo lo que el conjunto de drones hace. Entre las funciones de los *scripts* se incluye partes de la máquina de estados que se usa en la inteligencia artificial que se explicará más adelante.

En la función *Awake*, cuyo funcionamiento ya se ha explicado en el punto anterior, se instancia el modelo del dron *master* y los drones *slaves*. Para esto se obtiene por parámetros el número de drones que queremos para la simulación y los modelos. Una vez hecho esto los posicionaremos a dos metros del dron *master* generando una circunferencia. Esta se partirá ángulos de sesenta grados, como se extrae mediante la división de los grados de un circulo entre el número de drones, empezando a partir del eje central del dron principal, obligando a los drones a estar en posiciones equidistantes.

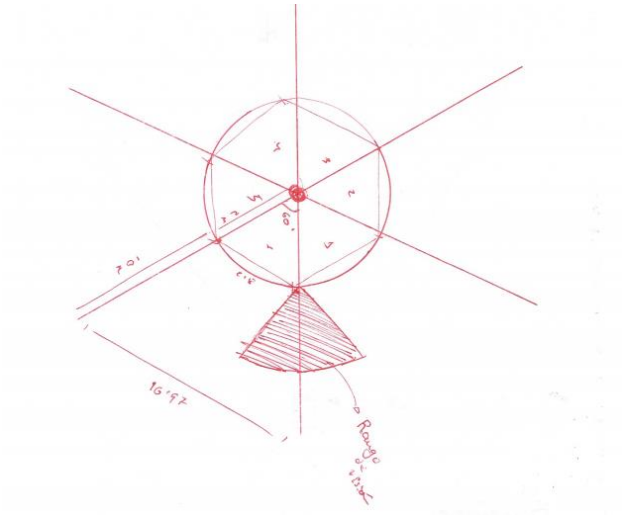


Ilustración 22. Boceto de la posición de los drones respecto al dron principal, se puede ver las distancias calculadas y el rango de visión de los drones pequeños.

También se añaden unos componentes a los drones *slaves* para que puedan moverse, estos componentes se llaman *CharacterController* y *NavMeshAgent* que brindan una serie de funciones, para el objeto al que ha sido añadido, permitiéndole moverse, detectar colisiones, etc.

La manera en la que se ha modelado los drones permite dividir y obtener todas las partes que lo componen por separado, facilitando la modificación de cualquier parte sin tener que modificar el resto. Para aplicar texturas obtenemos todas las partes del dron con la función *GetChild*, que nos devuelve una parte del objeto. A esta parte se le añade un elemento *TextRender* que permitirá asociar una imagen o textura, que hemos obtenido de la lista pública creada al principio del *script,* al objeto en cuestión.

Para aumentar el realismo de los drones se procede a generar unas animaciones desde la herramienta *Animation* que se encarguen de hacer que las hélices se muevan. Para ello asociamos los elementos de rotación de cada una de las hélices y modificamos las curvas que muestra la aplicación hasta que muestre un resultado adecuado.

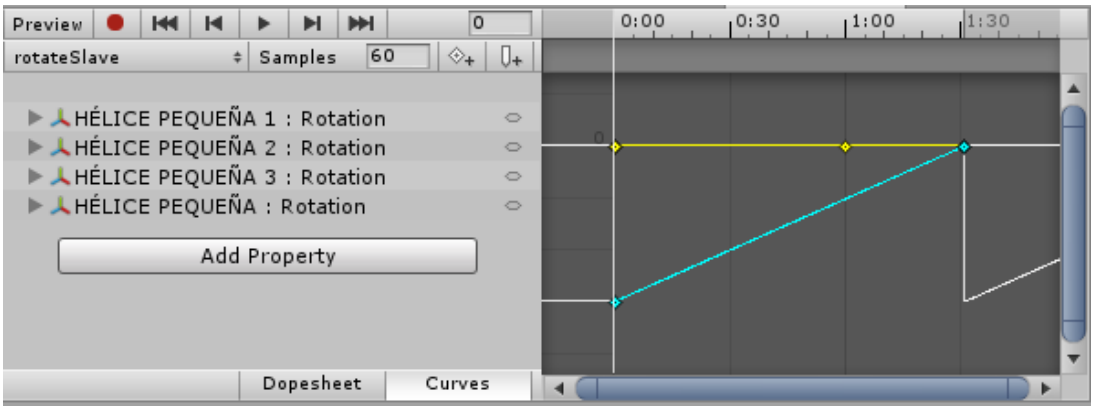


Ilustración 23. Representación de la animación de las hélices de los drones slaves

Posteriormente asociamos en el *script* al *GameObject* del dron la animación que acabamos de crear, le indicamos el *wrapmode*, que es la variable encargada del modo de reproducción del clip, que se indicara en modo bucle ya que es algo que se va a ejecutar siempre.

Se desarrolla un *script* llamado *Patrolling* que se encarga crear el objetivo o persona e instanciarla en una posición aleatoria, totalmente desconocida por el conjunto de los drones y que también se encarga de generar las pistas que deja ese objetivo.

Para la generación de las pistas, se recibe como parámetros el modelo de una bandera, a la cual se le aplica una textura a discreción del autor para mejorar la visibilidad de las pistas. Estas se generan en orden desde la posición en la que se ha instanciado el objetivo, también pueden aparecer en cualquiera de los puntos cardinales respecto al punto anterior, excepto si ya hay una pista o el objetivo.

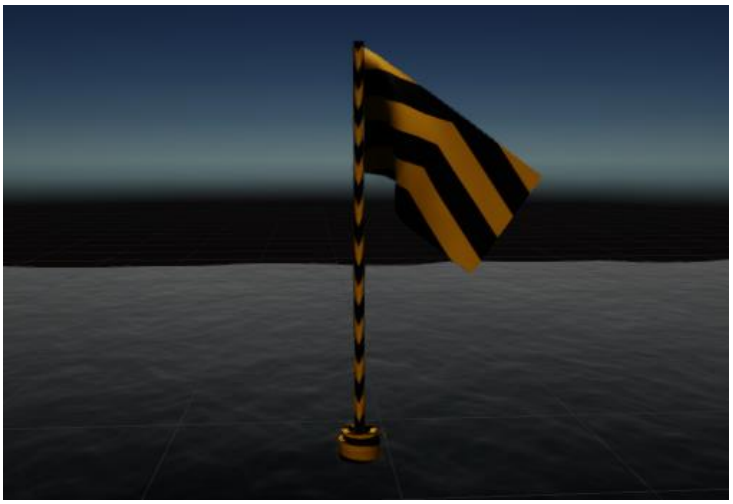


Ilustración 24. Ejemplo de bandera que se usa para indicar las pistas que va dejando la persona desaparecida.

A estos objetos se les añade un componente llamado *CapsuleCollider*, el cual genera una estructura invisible en forma de capsula que sirve para detectar las colisiones con otros objetos y viceversa. También se le aplican dos componentes más que se complementan: el *CharacterController* y el *RigidBody.* Al unirse evitan que los drones colisionen con el terreno que les envuelve.

En este *script* se desarrolla una parte importante, la guía y el recorrido que hace el conjunto de drones para patrullar todo el terreno sin dejarse ningún hueco, generando para ello puntos geo-posicionados en el mapa creado. Cuando se inicia la simulación se indica al sistema de navegación de los drones un punto desde el cual se inicia todo el recorrido y cuando estos llegan a un punto se cambia el destino al punto siguiente, y así hasta que se encuentre algún indicio o el objetivo.

Ilustración 25. Recorrido propuesto por este trabajo.

5.2.3 La Inteligencia artificial

A la hora de implementar la inteligencia artificial (IA) nos encontramos con varias posibilidades. La primera es el uso de *assets* para el desarrollo de la IA, lo que nos aporta máquinas de estado prefabricadas y múltiples funciones que pueden ayudar a la hora del desarrollo de nuevas máquinas y la segunda es el desarrollo propio, que es un poco más costoso, pero permite una personalización completa de la máquina que se crea.

Después de ver el diseño de las máquinas de estado en el apartado anterior y con las facilidades que nos da *Unity* se decide desarrollar de manera propia las máquinas y los eventos. Para el desarrollo de los eventos se va a hacer uso de los *Colliders*, contenidos por los objetos de la escena, y la función interna que permite detectar cuando hay una colisión con otro objeto. *Unity* brinda un conjunto de funciones para la detección de colisiones, las cuales reciben como parámetros el objeto con el que se colisiona. Estas funciones son *OnTriggerEnter, OnTriggerStay y OnTriggerExit* entre otras.

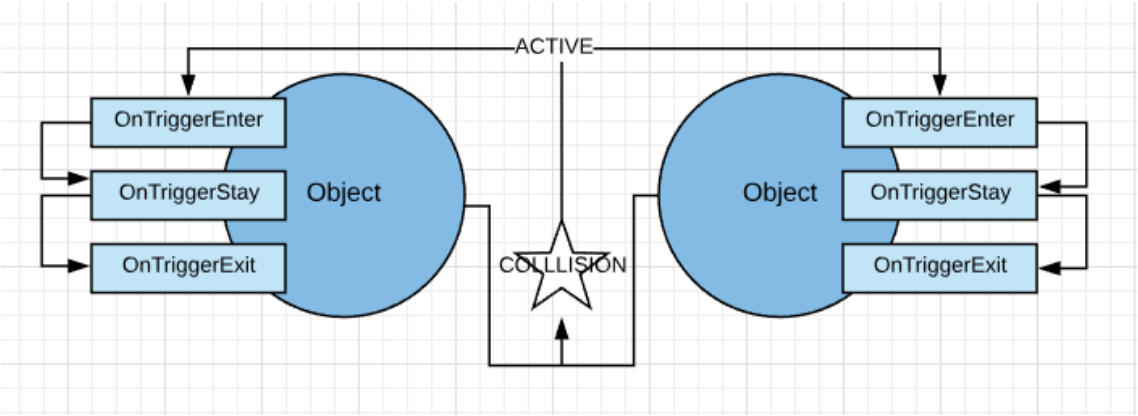


Ilustración 26. Ejemplo de reacción en una colisión.

Aquí es donde se desarrolla el código que se ejecuta cuando un objeto colisiona con otro, cuando un objeto mantiene la colisión y cuando un objeto deja de colisionar con otro objeto. Tanto el dron *master* como los drones *slave* tienen la primera función para detectar la colisión con las banderas o el objetivo principal. Esto ocurre gracias al componente *Collider,* al que se le ha aplicado como anchura o radio de la malla la profundidad visual de la cámara que tienen integrada, simulando la visión de estos.

Al activarse los eventos con las colisiones de los objetos, se opta por etiquetar con las etiquetas “FLAG” para las banderas y “OBJ” para el objetivo, para que a la hora de comprobar con que objeto se colisiona, sea una tarea menos ardua. Por esto la función *OnTriggerEnter* del dron *master* empieza comprobando la etiqueta del objeto con el que colisiona. Si se encuentra

con una bandera o indicio, se desactiva el *script* de *Patrolling* evitando así que siga con su ruta normal y se modifica el destino de los componentes que permiten el movimiento del dron, como pueden ser *CharacterController* y *NavMeshAgent,* para que se acerquen a la bandera en cuestión. Si se encuentra con el objetivo hace los mismos pasos que cuando encuentra una bandera, pero además genera un texto en pantalla mostrando los segundos que se ha tardado en encontrarlo.

En el momento que hay una colisión con una bandera, se ejecuta la función *OnTriggerStay*, que mientras esté en la misma posición o a dos metros y medios de distancia como máximo, se ejecuta la animación de la búsqueda exhaustiva hasta tres veces para una mejor búsqueda.

Finalmente, la función *OnTriggerExit* se encarga de eliminar ese indicio y guardar la posición en memoria para evitar encontrarlo recurrentemente.

5.2.4 La interfaz

Para que la experiencia del usuario sea completa, se tiene que desarrollar una interfaz intuitiva y fácil. En este apartado se plasma los pasos a seguir y las configuraciones que se usan para la interfaz gráfica del usuario, que costa de tres pantallas. Estas pantallas o *canvas,* como se les llama en el editor de *Unity*, es un plano de dos dimensiones en el cual puedes añadir objetos como cuadros de texto, botones, etc.

Ilustración 27. Ejemplo de los check buttons que se implementan.

La primera pantalla, la que el usuario ve nada más empezar la simulación, es la única activa al comienzo. Consta de un cuadro de texto en la parte superior que indica el título del simulador y debajo hay una serie de *check button*s que te permiten elegir opciones individuales sin excluir las demás. En la parte más baja, está el botón que da comienzo a la simulación. Todos estos botones tienen asociadas una serie de funciones para habilitar una parte del simulador, si se desea.

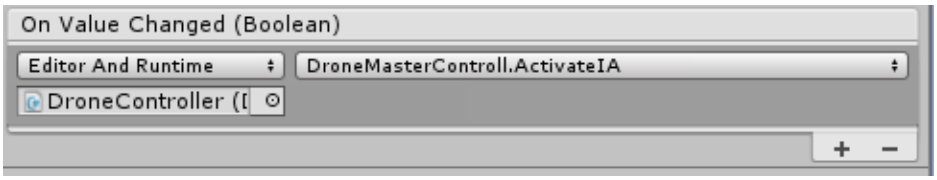


Ilustración 28. Ejemplo de la asociación de una función a el evento de cambio de valor.

En la parte más baja de esta pantalla podemos ver un pequeño cuadro negro con un nivel de transparencia bastante alto. En él se indica al usuario que para ver la ayuda debe mantener presionada la letra H.

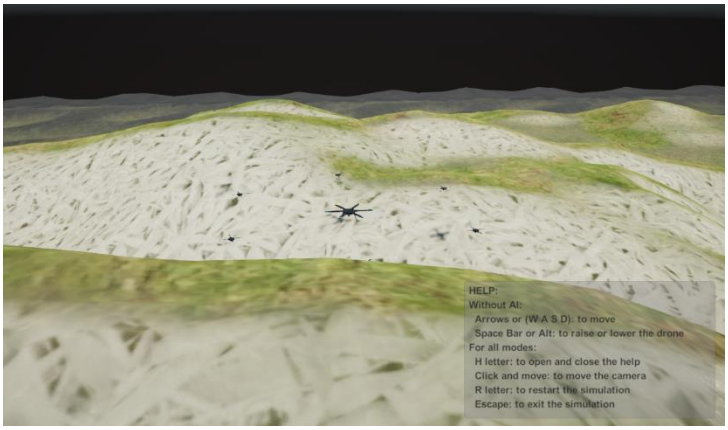


Ilustración 29. Ejemplo de la pantalla de ayuda durante la ejecución.

La segunda pantalla ya ha sido nombrada en el anterior párrafo y contiene la ayuda al usuario durante la simulación. Esta pantalla se crea, básicamente, con un objeto panel de color negro que tiene una transparencia del 80% y un texto de color blanco para que resalte en el fondo negro. Todo esto se activa mediante un evento controlado, el cual se llama *Input* y que tiene una función llamada *GetKey*, que verifica si la tecla que se le pasa por parámetros está pulsada o no.

Ilustración 30. Ejemplo de pantalla final.

La tercera pantalla o pantalla final es la que se muestra cuando los drones encuentran el objetivo. Se muestra la imagen de una bandera en la parte superior, la cual tiene activados los canales alfas para eliminar el fondo que contiene la imagen. También tiene un pequeño texto que contiene el tiempo que se ha tardado en realizar la simulación y dos botones, uno para reiniciarla y otro para cerrar la aplicación. La función asociada al reinicio se basa en recargar la escena desde el *SceneManager*, librería propia de *Unity* que aporta una gran cantidad de funciones para configurar la escena, así como funciones para pararla o reiniciarla, y la función asociada al botón que se encarga de cerrar el simulador usa el paquete de *Application*, propio de *Microsoft* y que lo incluye las librerías de *C#*.

5.2.5 La cámara

La implementación de la cámara aporta una visión periférica de la zona más cercana al dron, esto es importante para poder ver su comportamiento. El script que la gestiona empieza posicionando la cámara a una distancia de dos metros y a una altura de un metro más que el dron, gracias a esto y a la función *lookat*, se puede observar directamente el dron desde la distancia indicada, ya que la función indicada orienta el objetivo de la cámara hacia el objeto que se pasa por parámetros, en este caso el dron *master*.

Para los giros de la cámara, se tiene que captar el movimiento del ratón mediante la función *GetAxis*, que nos permite obtener el valor del movimiento. Estos valores, multiplicados por la velocidad a la que se quiera mover la cámara, se aplican a los vectores de rotación de la cámara respecto a los ejes X e Y, consiguiendo un movimiento fluido e intuitivo.

Para finalizar se mejora el control de la cámara, permitiendo mover la cámara solo con un botón del ratón presionado. Para ello se usa la función *GetMouseButtonDown* que permite activar o desactivar un pequeño *flag* para controlar las acciones que se realizan. Finalmente, se captura el movimiento de la rueda de *scroll* para acercar o alejar la cámara de la posición del dron y se aplica a la distancia o campo de visión, aumentado la visión periférica o disminuyéndola dando la sensación de *zoom*.

6. Pruebas

En este apartado se expone el simulador desarrollado, así como las pruebas reales realizadas por un dron real en una zona urbanizada y los resultados obtenidos, para poder determinar la viabilidad del proyecto.

El principal resultado de este proyecto es el simulador en sí, el cual después de varias ejecuciones arroja una serie de resultados que muestran el tiempo de todas las pruebas, lo que permitirá determinar si el algoritmo es viable.

Ilustración 31. Imagen de inicio del simulador.

En la ilustración anterior podemos ver como ya está creado todo el entorno gracias a la programación de las funciones *Awake*. En el primer momento en el que se utiliza esta pantalla, se puede observar que no funciona nada excepto las opciones que se ven en la imagen. Una vez se prueban todas las configuraciones y se presiona el botón de inicio, la simulación comienza y se ejecuta hasta que finaliza tras encontrar el objetivo.

Durante estas pruebas se observan varios problemas, siendo el principal mejorar el rendimiento gráfico del simulador para quitarle peso a la tarjeta gráfica. Estas mejoras se hacen aplicando un sistema de post-procesado, aplicando filtros y efectos en la pantalla que mejora drásticamente el rendimiento.

Ilustración 32. Configuración del sistema de post-procesado de imagen.

Como se observa en la ilustración 32, la primera configuración activada es la llamada *fog*. Esta opción genera una capa de color delante de los objetos dependiendo de la distancia, evitando renderizar elementos lejanos.

Para evitar la carga de partes del terreno a los que la cámara no apunta, se activa las opciones de configuración *Occluder Static* y *Occludee Static* en todos los objetos deseados. Estas opciones, a la hora de activar el *scene filter* de la ventana de configuración *Occlusion* de *Unity,* permiten eliminar los objetos que no estén visualizados por la cámara, mejorando el rendimiento de manera drástica.

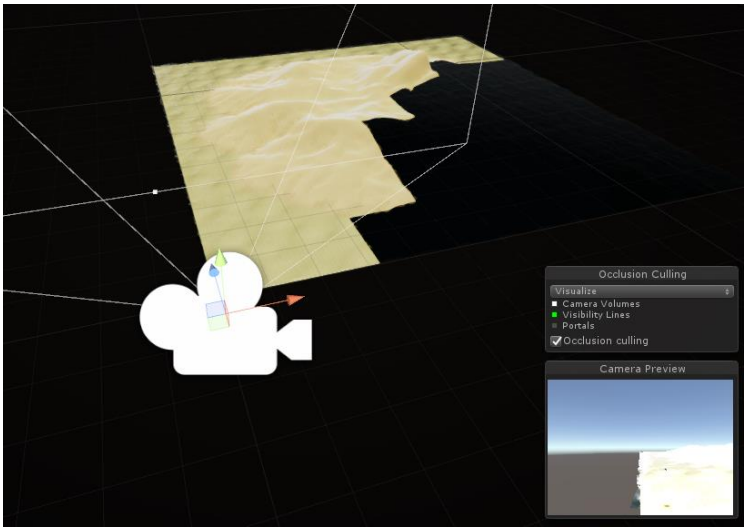


Ilustración 33. Ejemplo de occlusion culling.

Por otra parte, la opción llamada *antialiasing* no mejora el rendimiento gráfico sino lo contrario, puesto que mejora el bordeado de los objetos para una mejor visualización. La

opción elegida permite mejorar el aspecto de los objetos más próximos antes. Las diversas pruebas hechas muestran una verdadera mejora, puesto que se seleccionó un *preset* que mejorase el rendimiento y la calidad grafica se veía extremadamente afectada.

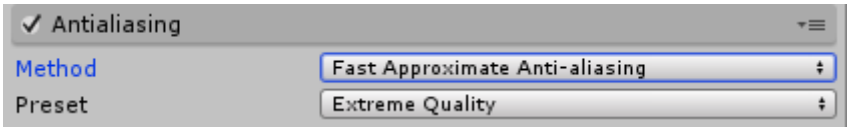


Ilustración 34. Configuración del antialiasing.

Las siguientes opciones mejoran la visualización del simulador sin empeorar el rendimiento gráfico. La opción *ambiente occlusion*, oscurece los elementos que están cerca entre ellos y no reciben luz focal directa. *Motion blur* es la opción que se encarga de emborronar los objetos a los que no sigue la cámara y depende siempre de la velocidad de la cámara.

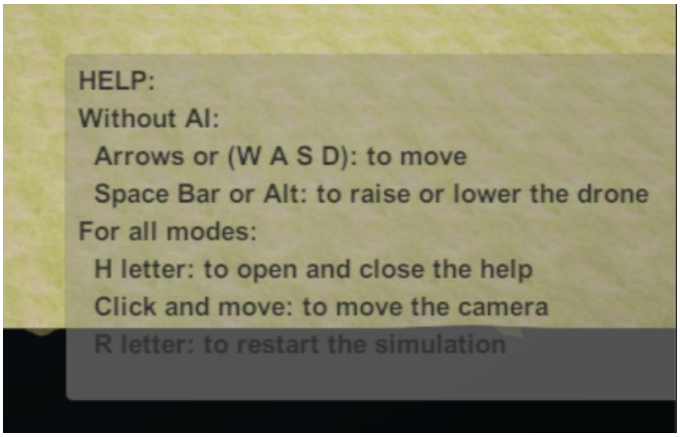


Ilustración 35. Ayuda por pantalla para el usuario

Para comprobar que los atajos de teclado funcionan, se prueba la ayuda por pantalla, como se puede observar en la imagen anterior. También se utiliza la tecla de *escape* y la letra 'R' para cerrar la aplicación y reiniciarla, funcionando según lo esperado.

A la hora de finalizar la aplicación, encontrando el objeto, se observa la última pantalla que el usuario ve. Esta pantalla contiene el tiempo total que se ha tardado en encontrar el objetivo y dos botones que funcionan sin problemas.

La realización de pruebas reales en campo abierto es sumamente importante para verificar la viabilidad del proyecto. Se implementa el algoritmo desarrollado en este trabajo en un dron real prestado, arrojando unos resultados excelentes que verifican la viabilidad del proyecto.

Ilustración 36. Imagen del dron real usado para las pruebas. Fuente: www.yuneec.com

En la imagen anterior se puede observar el dron usado para las pruebas en campo abierto. Este dron es de la marca *Yuneec*, es el modelo *Typhoon H*. Posee una cámara de alta calidad, con una resolución 4K que puede rotar 360º en los 3 ejes, entre otras características.

Las pruebas de vuelo se realizaron en un barrio a las afueras de Asunción, capital de Paraguay. El vuelo se realizó en un terreno de 600m de ancho y 700m de largo, a una velocidad media de 4m/s por segundo, en alrededor de veinte minutos y como resultado se obtuvieron ciento cincuenta imágenes georreferenciadas.

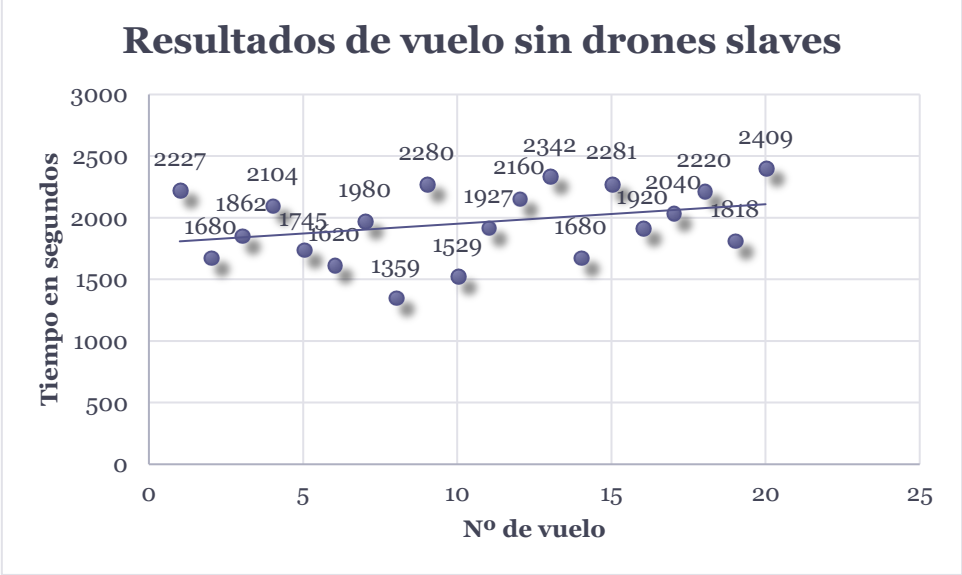
Ilustración 37. Ejemplo de imagen generada por el dron.

Estas imágenes tratadas con un programa especializado, como es el *Pix4D,* se puede obtener un modelo tridimensional de la zona que se ha sobrevolado, permitiendo detectar objetos y personas que, posiblemente, no se hubiesen visto a simple vista o a pie de calle.

Ilustración 38. Resultado del tratamiento de las imágenes.

Como bien se observa en la imagen, la creación de una malla tridimensional no genera unos resultados perfectos, pero posibilita la detección de personas y objetos de una forma que no se podría conseguir a pie de calle.

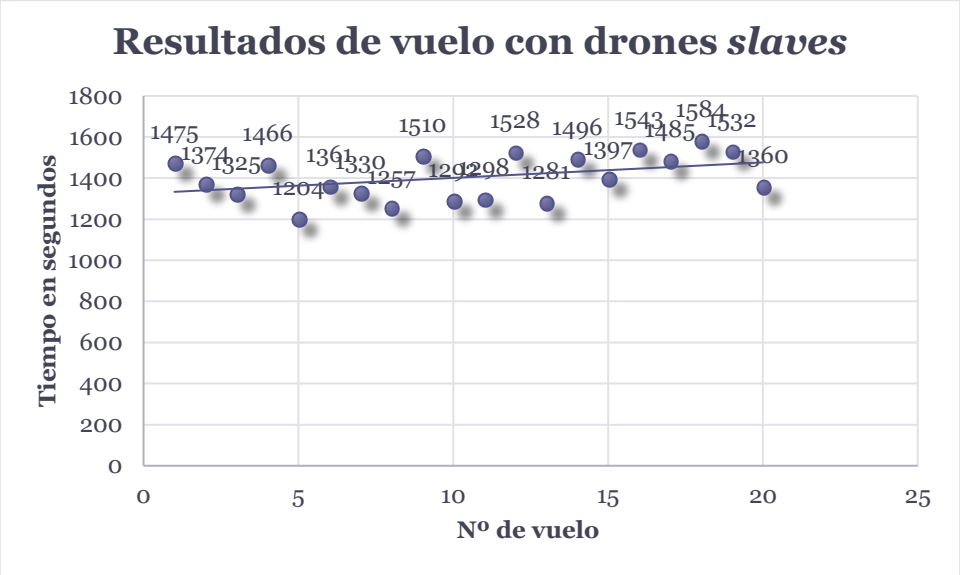
Ahora se exponen los resultados obtenidos de veinte vuelos diferentes. Estos vuelos se han hecho desde el editor de *Unity* con una opción especial que le permite hacer múltiples vuelos y guardarse los resultados para mostrarlos posteriormente.



Gráfica 1. Resultados son drones slaves.

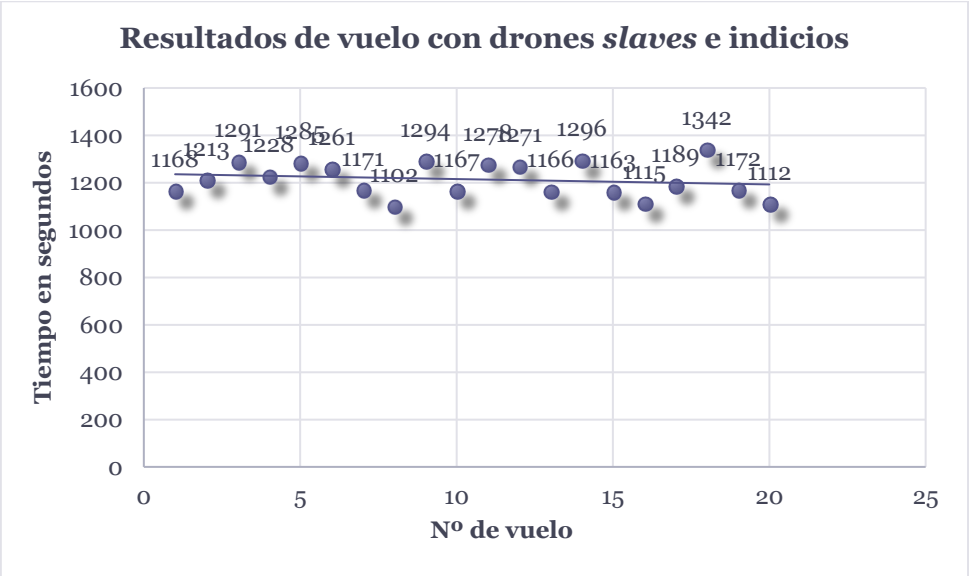
En la gráfica 1 se muestran los datos de veinte vuelos, estos vuelos se han realizado sin el uso de los drones *slave*. Como se puede observar hay gran variedad de resultados, pero si se

observa la línea de tendencia, se puede intuir que la media de los datos ronda los dos mil segundos, aproximadamente 33 minutos.



Gráfica 2. Resultados con drones slaves

En la gráfica anterior, se muestran los datos relacionados a los vuelos con drones *slaves*, si se observa con detenimiento los datos arrojados muestran una media aproximada de mil cuatrocientos o aproximadamente 23 minutos. Esto muestra que la visión extra que aportan los drones es determinante para reducir el tiempo de búsqueda de la persona perdida.



Gráfica 3. Resultados con drones slaves e indicios.

Se pueden observar en la gráfica 3 los resultados arrojados por la prueba compuesta por los drones *slaves* y los indicios. Estos resultados mejoran en doscientos segundos el resultado anterior, dejando la media en aproximadamente 20 minutos.

7. Conclusiones

De entre todos los motores gráficos del mercado, se seleccionó *Unity* debido a su gran accesibilidad a contenidos gratuitos, versatilidad de uso y la utilización de un paradigma orientado a objetos que le otorga un gran manejo de todas las herramientas disponibles al programador. La gran ventaja que tiene el programador de la aplicación es haber aprendido varios lenguajes orientados a objetos en el grado cursado, entre ellos *C#,* ya que se usa en el *scripting* de *Unity*.

Por otra parte, el uso de V*isual Studio*, con el cual el programador ya tiene experiencia previa, ha resultado ser muy útil tanto para la parte de desarrollo de *scripts* como la depuración. *Visual* tiene una opción para asociar la ejecución del código con la ejecución de prueba del editor de *Unity,* esto permite añadir puntos de parada en el código para evaluar el valor de las variables en cierto momento de la ejecución. Por eso la elección de *Visual Studio* respecto a *Mono Develop* ha resultado, finalmente, beneficiosa.

Uno de los objetivos planteados al inicio del documento, proponía la creación de un terreno procedural con ciertas características. Los algoritmos que desarrollan este terreno han resultado más complejos de lo esperado, sobre todo el que se encarga de generar una matriz de datos respecto a la imagen dada. La transformación de los datos obtenidos a un objeto *Texture* no siempre era la deseada y la información recolectada no arrojaba luz al problema. La solución al problema expuesto es la modificación de la entrada de los datos y la posterior conversión de estos, permitiendo así la perfecta creación de la matriz.

Otro de los objetivos propuestos era el diseño de un dron que se ciñese a las características que el entorno exigía. Para eso el alumno se informa de las posibles características que se pueden necesitar a la hora del modelado de un dron. Esta información es insuficiente puesto que, a la hora de comparar el dron modelado con otros drones, el desarrollado por el alumno carece de protectores para las hélices, por lo que cabe la posibilidad de que sea demasiado fino para la circuitería que tiene que contener y la envergadura sea más bien pequeña respecto a drones reales.

Cuando se evalúa la conformidad con el objetivo que habla sobre la navegación hay diversidad de opiniones. El alumno que ha realizado el trabajo opina que es un sistema de navegación útil y fiables para comprobar cómo debería funcionar la simulación y los problemas que tiene el dron. Por otra parte, para una persona que haya volado habitualmente un dron, puede pensar

que esta navegación está limitada, puesto que en la mayoría de los casos no se puede controlar un dron desde una vista en tercera persona, más bien se puede controlar desde el dispositivo radiocontrol con la imagen de la cámara a bordo.

La navegación ha resultado ser otro problema tanto a la hora de desarrollar los *scripts* de la cámara como las funciones de movimiento del dron. Estos problemas se deben a la posición de los ejes locales de los objetos, puesto que no se debe determinar que el eje de la cámara es el mismo que el del conjunto de drones ya que no siempre es así. Por otra parte, para que la cámara pueda rotar según el movimiento del ratón, se usa una función llamada *LocalRotate,* que permite la rotación alrededor de un punto dado, este problema es el mismo que el expuesto anteriormente.

Y para finalizar con los objetivos, tenemos el desarrollo de la inteligencia artificial. La idea principal era el uso de un *asset* gratuito para la construcción de las máquinas de estado. Este *asset* disponía de una opción para desarrollar las maquinas como si fuesen diagramas, haciendo más visible el diseño de estas. El problema surgió cuando eliminaron el *asset* de la tienda de *Unity* dejando sin soporte la aplicación. Puesto que el desarrollo y el diseño de unas máquinas de estado como las que se implementan no necesita de conocimientos extensos sobre el tema, finalmente se implementan de manera propia.

Para resumir los resultados, se va a hacer una pequeña comparación a modo de conclusión sobre estos. Se estima que una persona humana se mueve a la misma velocidad que el dron sobre el terreno y que esa velocidad se reduce cuando encuentra obstáculos o situaciones en los que transitar el sitio a patrullar es prácticamente imposible. Recorrer, para esa persona, un espacio como el proyectado en la simulación, puede costar entre dos y cinco horas; ahora, si nos ceñimos a los resultados obtenidos por los diferentes vuelos, podemos comprobar que la reducción del tiempo es sustancial, ya que la búsqueda más larga realizada por un dron en las simulaciones es de 2409s que aproximadamente son 40 minutos y como mínimo 1102s que son aproximadamente 18 minutos. Cabe destacar que el tiempo máximo estimado para el uso de una batería de dron es de 20 minutos por lo que, tardando entre diez y quince minutos en cambiar la batería por una llena, la primera opción se transforma en 55 minutos, que sigue siendo menos que el mínimo tiempo en el que una persona puede recorrer el terreno proyectado.

Como final, se destaca la gran labor de aprendizaje del alumno en las herramientas utilizadas, ya que no tenía experiencia previa en ninguna de ellas. Esto ha ocasionado un gran trabajo de investigación, principalmente sobre *scripting* en *Unity* y también sobre los demás programas

7.1 Relación del trabajo desarrollado con los estudios cursados

El alumno que ha desarrollado este proyecto ha estudiado el Grado en Ingeniería Informática por la Universidad Politécnica de Valencia, con especialización en la rama de computación. Esta rama capacita para diseñar sistemas informáticos complejos, teniendo en cuenta criterios de eficiencia, fiabilidad y seguridad. Prepara para ser capaz de evaluar estos requerimientos y recomendar los lenguajes de programación y métodos algorítmicos más adecuados para cada problema. Las asignaturas de sistemas gráficos incluidas en la rama le permiten al alumno entender con mayor facilidad muchos de los términos y aspectos utilizados en el editor seleccionado para este trabajo.

Los estudios cursados han sido esenciales para el buen desarrollo de este proyecto. No solo las clases teóricas y prácticas recibidas por los profesores, sino también las metodologías usadas y las competencias transversales que, aunque recién implementadas, han mejorado la resolución de problemas.

8. Trabajo futuro

El trabajo desarrollado sirve de inicio para un proyecto más grande. Las mejoras que se pueden realizar sobre este proyecto son varias:

- Generar una cámara que se posicione en el mismo lugar que las cámaras que llevan acopladas los drones dentro de la simulación para poder ver lo que se observa desde ahí.
- Aplicar un *collider* a la cámara con la forma del campo de visión que realmente tendría en la vida real.
- Aplicar vegetación a la simulación, lo cual no se ha hecho por falta de recursos físicos.
- Dar la opción de elegir entre varios tipos de terreno con características diferentes.
- Adecuar el modelado de los drones a las características de las diferentes búsquedas.
- Permitir la diversidad de situaciones horarias en el simulador. No es lo mismo hacer una búsqueda a medio día que por la noche.
- Permitir la ejecución en paralelo de diferentes vuelos para recabar información con mayor velocidad.

Estos puntos buscan mejorar el proyecto realizado, dando una cobertura mayor y sirviendo para una solución real.

9. Referencias

[1] SOBRE LA ACTUACIÓN POLICIAL ANTE LA DESAPARICIÓN DE MENORES DE EDAD Y OTRAS DESAPARICIONES DE ALTO RIESGO. http://www.lamoncloa.gob.es/Documents/bfd7-dc50-instruccion_1_2009_ses_desaparecidos.pdf

[2] Molina, P., Colomina, I., Vitoria, T., Freire, P., Skaloud, J., Kornus, W., … Aguilera, C. (2011). THE CLOSE-SEARCH PROJECT: UAV-BASED SEARCH OPERATIONS USING THERMAL IMAGING AND EGNOS-SOL NAVIGATION. Retrieved from http://www.isprs.org/proceedings/2011/Gi4DM/PDF/OP10.pdf

[3] Búsqueda y localización de personas desaparecidas. http://www.policiacanaria.com/sites/default/files/busqueda-localizacion-personas-desaparecidas.pdf

[4] Wikipedia – Vehículo aéreo no tripulado. https://es.wikipedia.org/wiki/Vehículo_aéreo_no_tripulado

[5] ¿Drone Ala Fija o Multirotor? ~ ICGeo Ingeniería Civil y Geología. http://www.icgeo.org/2018/01/es-mejor-un-drone-ala-fija-o-multirotor.html

[6] Navarro, J. A., Parés, M. E., Colomina, I., & Blázquez, M. (2017). A Generic, Extensible Data Model for Trajectory Determination Systems. In *Proceedings of the 3rd International Conference on Geographical Information Systems Theory, Applications and Management* (pp. 17–25). SCITEPRESS - Science and Technology Publications. https://doi.org/10.5220/0006258400170025

[7] Del Estado, J. (2014). Disposición 10517 del BOE núm. 252 de 2014. http://www.seguridadaerea.gob.es/media/4389070/ley_18_2014_de_15_octubre.pdf

[8] Computer Graphics with Open GL. https://doc.lagout.org/programmation/OpenGL/Computer Graphics with OpenGL %284th ed.%29 %5BHearn%2C Baker %26 Carithers 2013%5D.pdf

[9] Federación Española de Deportes de Montaña y Escalada. http://fedme.es

[10] Terrain: Introduction to Heightmaps - Unity. https://unity3d.com/es/learn/tutorials/topics/graphics/terrain-introduction-heightmaps

[11] INFORME ENERO 2017 SOBRE PERSONAS DESAPARECIDAS EN ESPAÑA. http://www.interior.gob.es/documents/10180/6960463/Informe_Desaparecidos_España_2017.pdf/16f1015e-c91a-460a-9f55-059bb970a0cb

[12] Understanding Steering Behaviors: Wander. https://gamedevelopment.tutsplus.com/tutorials/understanding-steering-behaviors-wander--gamedev-1624

[13] Bringing 3D terrain to the browser with Three.js. (n.d.). Retrieved from https://blog.mapbox.com/bringing-3d-terrain-to-the-browser-with-three-js-410068138357

[14] Unity: Animation. https://www.youtube.com/watch?v=gaKDttsEwCs

[15] Generating Terrain from Heightmaps in Unity 3d. https://www.youtube.com/watch?v=VLTBYmmvLus

[16] Terrain: Introduction to Heightmaps. https://unity3d.com/es/learn/tutorials/topics/graphics/terrain-introduction-heightmaps

[17] Tutorial 3: Generating and rendering a terrain height map. http://libnoise.sourceforge.net/tutorials/tutorial3.html

[18] Terrain Blending Starter Kit. https://assetstore.unity.com/packages/vfx/shaders/terrain-blending-starter-kit-91921

[19] Tipo de drones en función del control. http://www.areatecnologia.com/aparatos-electronicos/drones.html

[20] Foro Drones. http://www.forodrones.com/

[21] Unity Scripting. https://unity3d.com/learn/tutorials/s/scripting

[22] Create a Simple Terrain Level in Unity. https://www.youtube.com/watch?v=GBtrC3a0xBA

[23] MundoDron. https://foro.mundodron.net/