Desarrollo de una plataforma basada en Unity3D para la aplicación de IA en videojuegos

Óscar Manuel Losada Suárez, Víctor Rodríguez-Fernández, Cristian Ramírez-Atencia, and David Camacho

Universidad Autónoma de Madrid
oscarlosu@gmail.com,
{victor.rodriguez,cristian.ramirez}@inv.uam.es,
david.camacho@uam.es
AIDA (Applied Intelligence & Data Analysis): http://aida.ii.uam.es/

Abstract. La utilización intensiva de diferentes técnicas relacionadas con la Inteligencia Artificial (IA) en el área de los videojuegos ha demostrado ser una necesidad para el campo. El uso de estas técnicas permite dotar de una mayor flexibilidad y adaptabilidad a los juegos que es muy apreciada por los jugadores. Temas como la generación procedimental de contenido, la creación de agentes que puedan jugar a un videojuego de forma competente, o de agentes cuya conducta sea indistinguible de la de un jugador humano atraen a una cantidad creciente de investigadores. El objetivo de este trabajo es la presentación de una plataforma basada en el motor Unity3D que permita de manera simple la integración y prueba de algoritmos de IA. La plataforma ofrecerá como nuevas características, adicionales a las ya disponibles en la actualidad, la utilización de un entorno 3D, el desarrollo de un juego innovador (basado en múltiples agentes), y la exploración de aspectos de juego como el análisis del terreno, la cooperación entre agentes independientes y heterogéneos, la comunicación de información entre los mismos y la formación de jerarquías.

Keywords: Videojuegos, Agentes Inteligentes, Plataforma Software, Unity3D, Inteligencia Artificial

1 Introducción

La investigación de la Inteligencia Artificial en los videojuegos comenzó en torno al juego *PacMan*, especialmente en su versión *Ms. Pac-Man*, después de que investigadores como J. Koza y J. Rosca lo propusieran como un entorno interesante que planteaba un problema de priorización de tareas y permitía la aplicación de algoritmos genéticos [13]. El interés de la comunidad investigadora por la investigación en los videojuegos ha ido en aumento desde entonces, llegándose a celebrar competiciones para desarrollar agentes inteligentes que jueguen a *Ms. Pac-Man* [5] y a *Infinite Mario Bros* [16] en los últimos años.

El objetivo de este trabajo es crear una plataforma para el desarrollo y prueba de algoritmos de Inteligencia Artificial sobre el entorno Unity3D. Las principales contribuciones que ofrece sobre las plataformas existentes en la actualidad incluyen la utilización de un entorno 3D, el desarrollo de un juego innovador (basado en múltiples agentes) y la exploración de aspectos de juego como el análisis del terreno, la cooperación entre agentes independientes y heterogéneos, la comunicación de información entre los mismos y la formación de jerarquías.

El juego en el que se basa la plataforma consiste en un juego de acción táctica en 3D con dos equipos integrados por varios jugadores, cada uno de los cuales será controlado por un agente de Inteligencia Artificial. Más concretamente, el juego está basado en el deporte *Paintball*, en el que los jugadores deben tratar de marcar a sus oponentes con bolas de pintura disparadas con pistolas de aire comprimido.

La intención de este proyecto es que pueda ser utilizado por investigadores y desarrolladores de Inteligencia Artificial de forma semejante a como se han usado otros juegos en eventos como el IEEE CIG (Computational Intelligence and Games) [7] o en CEC (Conference on Evolutionary Computation) [12].

El resto del trabajo se estructura de la siguiente forma: La sección 2 hace un breve resumen de los principales focos de investigación de inteligencia artificial en videojuegos. En la sección 3 se resume el diseño y la arquitectura de la plataforma y sus dos partes principales: el generador de mapas y el juego. La sección 4 recoge los resultados de la pruebas experimentales realizadas con una serie de agentes de IA implementados como ejemplo. Finalmente, la sección 5 muestra las conclusiones y propone posibles líneas de trabajo futuras.

2 Estado del Arte

En el ámbito de la Inteligencia Artificial aplicada a videojuegos, tradicionalmente se han explorado tres vías:

- *Jugadores inteligentes*: Se dedica a crear agentes que reciben información del estado del juego y del entorno a través de sensores, evalúan dicha información y actúan en función de ella a través de actuadores que pueden afectar al estado del juego.
- *Creación procedimental de contenido*: Consiste en desarrollar algoritmos que generen contenido, generalmente niveles o mapas, que se consideren "interesantes".
- *Test de Turing*: Trata de crear un agente que juegue a un juego de forma que sea indistinguible de un humano jugando.

Dentro de la categoría de creación de jugadores inteligentes, en la que se centra este trabajo, las plataformas de inteligencia artificial desarrolladas a día de hoy se basan en juegos clásicos como *Ms. Pac-Man* en *Ms. Pac-Man AI Competition* y *Ms. Pac-Man vs Ghosts Competition* [1], Infinite Mario Bros. en *Mario AI Competition* [17], y *StarCraft* en *StarCraft AI Competition* [7]. Por otro lado, en la actualidad abundan las plataformas basadas en juegos serios, cuyo objetivo

no es sólo la diversión sino también el entrenamiento o aprendizaje de una serie de habilidades. En este ámbito podemos encontrar desde agentes inteligentes de aprendizaje en mundos virtuales [2] hasta entornos de entrenamiento gamificados para operaciones con drones [11, 10].

Típicamente, el desarrollo de jugadores inteligentes utiliza técnicas clásicas de Inteligencia Artificial tales como máquinas de estado finitas [15] y árboles de comportamiento [9], aunque en los últimos tiempos también se aplican algunas más sofisticadas como algoritmos evolutivos [6], algoritmos de colonias de hormigas [3] y métodos de Monte Carlo [14].

De cara a clasificar los juegos anteriormente mencionados a la hora de definir el tipo de problema que plantean a los agentes, uno de los miembros del equipo desarrollador de la plataforma *CodinGame*[1] propone las siguientes características:

- *Conocido* o *Desconocido*, dependiendo de si se sabe a priori como va a reaccionar el entorno a las acciones del agente.
- *Accesible o inaccesible*, dependiendo de si los agentes tienen acceso a toda la información del entorno que podría usarse para tomar decisiones.
- *Determinista o no determinista*, dependiendo de si el comportamiento del entorno esta definido lógicamente o probabilísticamente.
- *Estático o dinámico*, dependiendo de si el entorno cambia mientras que los agentes están tomando decisiones.
- *Discreto o Continuo*, dependiendo de si hay un número finito o infinito de acciones posibles para el agente.
- *Solitario o multijugador*, dependiendo de si intervienen uno o más agentes en el juego.

Para el desarrollo del proyecto se ha usado principalmente el motor *Unity3D*[2]. Los motivos para la elección de este motor de juego son múltiples: Es multiplataforma, tiene una completa librería con soporte para *rendering*, sonido, físicas y controles, y existe una enorme comunidad de usuarios dispuesta a compartir su conocimiento.

3 Desarrollo de la plataforma IA sobre el motor Unity3D

En este apartado se describe el diseño de la plataforma (ver Figura 1), la cual se ha dividido en dos partes: el generador de mapas y el juego. Este segundo, a través de una API usada para definir los personajes controlados por los usuarios, permite la introducción de agentes inteligentes.

La plataforma está enteramente desarrollada sobre Unity3D, que proporciona un sistema de navegación para un entorno 3D. Este sistema permite crear de forma automática una malla de navegación (*NavMesh*) a partir del conjunto de meshes que definan el mapa y utilizándola, proporciona funciones que automatizan la navegación entre dos puntos cualesquiera del mapa.

[1] Codingame: https://www.codingame.com/start
[2] Unity3D: https://unity3d.com/es

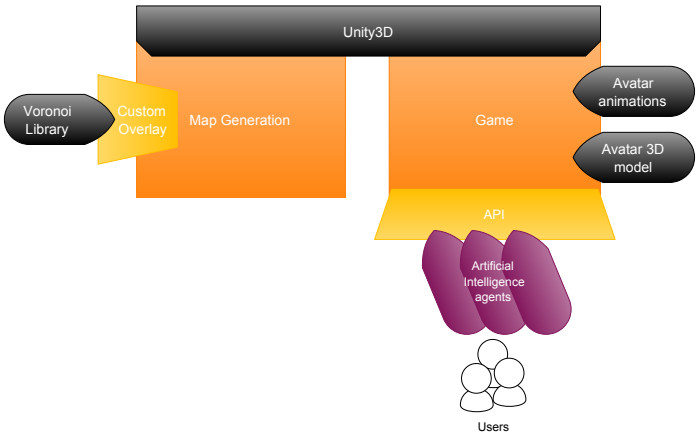


Fig. 1: Esquema de la arquitectura de la plataforma.

3.1 Juego

PaintBol, el videojuego desarrollado en este trabajo como parte de la plataforma, es un juego de acción táctica de cero jugadores (sin necesidad de jugadores humanos, en el que las IAs pueden enfrentarse entre ellas), basado en el deporte *Paintball*, donde compiten dos equipos de cinco jugadores. El funcionamiento básico del juego se puede ver en la Figura 2. La clase MatchHandler se encarga de proporcionar a las instancias de la clase Actor, que representan a los personajes del juego la información que les proporcionan sus sentidos o sensores (vista y oído) y por otra parte se encarga de efectuar las acciones de los personajes sobre su entorno. Así, las partidas consisten esencialmente en un bucle en el que se alterna la recolección de información para proporcionar a las IAs que controlan a los personajes del juego con la cesión del control a dichas IAs para que analicen la información de que disponen y tomen decisiones.

En este juego los sensores del jugador son la vista y el oído. Con la vista los jugadores reciben información sobre otros jugadores y objetos del terreno (objetivos visuales) en su linea de mira (arco frontal de 120 grados y alcance de 50 unidades) siempre que no haya otro objeto obstruyendo la recta de visión al objetivo visual. Por otro lado, cada acción en el juego produce cierta cantidad de ruido que puede ser oída por los jugadores si están lo suficientemente cerca de la fuente de ruido cuando este se produce.

Por otro lado, las acciones que pueden realizar los jugadores son: movimientos (correr, andar, andar agachado), comunicación (compartición de información con otros jugadores), disparar a enemigos en el rango de vista, ataques en espacios cerrados (afectan a todos los jugadores en un arco frontal de 120 grados y 2 unidades de distancia) y lanzar granadas en una dirección horizontal y ángulo vertical (afectan a jugadores en un área de 5 unidades alrededor de donde cae).

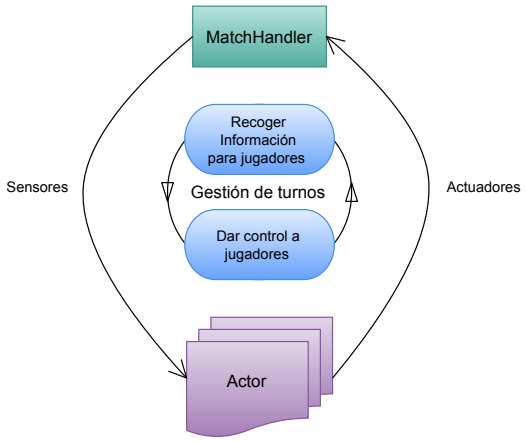


Fig. 2: Esquema de funcionamiento del juego.

En este juego cada jugador tiene 5 vidas. Cuando un jugador es alcanzado por un ataque, pierde una vida y desaparece durante 6 segundos, para luego reaparecer en un lugar aleatorio. Cuando el jugador pierde todas las vidas, pasa al estado "muerto" y no vuelve a aparecer.

El sistema de puntuación de *PaintBol* tiene en cuenta los siguientes parámetros:

- *Vidas quitadas*, que representa el número de veces que se ha alcanzado a enemigos con un ataque cualquiera.
- *Cuerpo a cuerpo*, que representa el número de veces que se ha alcanzado a enemigos con un ataque cuerpo a cuerpo.
- *Disparando*, que representa el número de veces que se ha alcanzado a enemigos con un disparo.
- *Granadas*, que representa el número de veces que se ha alcanzado a enemigos con una granada.
- *Con Sigilo*, que representa el número de veces que se ha alcanzado a enemigos con un ataque antes de que el enemigo viese al jugador.
- *A la cabeza*, que representa el número de veces que se ha alcanzado a enemigos con un disparo dirigido a la cabeza al apuntar.
- *Muertes múltiples*, que representa el número de veces que se ha conseguido alcanzar a más de un enemigo con un mismo ataque.
- *Fuego amigo*, que representa el número de veces que se ha alcanzado a aliados con un ataque.
- *Vidas restantes*, que representa el número de puntos de vida que conserva un jugador.
- *Granadas restantes*, que representa el número de granadas que conserva un jugador.

Finalmente, estos parámetros son utilizados para computar los *puntos por vidas quitadas*, *bonus por vida restante*, *bonus por granadas restantes*, *bonus por*

puntería y *bonus por variedad*. A su vez, la puntuación total de cada jugador se computa como la suma de estos parámetros, y la puntuación total de cada equipo como la suma de las puntuaciones totales de cada jugador del equipo.

Con respecto a la clasificación de videojuegos como plataforma de IA (Ver Sección 2), se puede apreciar que *PaintBol* es un juego de tipo *conocido* porque las reglas están preestablecidas y son conocidas por los agentes. Es *inaccesible*, porque los agentes tienen sólo información parcial del estado de la partida en cada momento. Es *no determinista* porque al comenzar la partida y cuando son alcanzados por un ataque, los jugadores aparecen en posiciones aleatorias del mapa y porque al disparar existe un factor aleatorio que influye en que el disparo alcance a su objetivo o no. El juego es *estático o dinámico* en función de como sean los algoritmos de los agentes: si todas las decisiones se toman de forma atómica en menos de un frame, entonces sería estático, pero los agentes pueden tener memoria y usar varios turnos para tomar una única decisión, en ese caso sería dinámico. Como el espacio en el que actúan los agentes es continuo hasta donde permite la capacidad de representación de los ordenadores, el juego es de tipo *continuo*. Finalmente, es *multijugador* por definición, pues el problema consiste precisamente en que varios agentes se enfrenten en el contexto de las reglas del juego.

El *entorno tridimensional* en el que se desarrolla el juego, junto con la relevancia de la topografía de los mapas y la distribución de los objetos del terreno para las mecánicas del juego hacen que el *análisis del terreno* (terrain reasoning) sea una herramienta útil e interesante para los agentes. Para aumentar las posibilidades de la plataforma, ésta cuenta con una herramienta de generación de mapas, que evita que se puedan diseñar estrategias basadas en las características específicas de mapas concretos y además implica que las características de los escenarios en los que se desarrolla la acción podrían redefinirse y ampliarse en el futuro. De los juegos mencionados en el estado del arte, sólo en *StarCraft* es tan relevante en este aspecto, pero en este caso los mapas son conocidos a priori.

Los equipos están formados por *agentes* posiblemente *heterogéneos* e *independientes*. Son heterogéneos en el sentido de que los algoritmos que controlan a cada uno de los agentes pueden ser completamente diferentes, y son independientes porque todos los agentes tienen información parcial no compartida del estado del juego en función de diferentes parámetros como su posición, la dirección en la que estén mirando, etc. Por ejemplo, los agentes en principio no saben donde están sus aliados ni sus enemigos, ni qué están haciendo a no ser que estén dentro de su arco de visión.

Estas características permiten que se formen estrategias basadas en jerarquías, que dan lugar a situaciones interesantes cuando, por ejemplo, parte de un equipo queda fuera de la partida y los agentes restantes podrían saber reaccionar a esa situación.

Los agentes tienen la capacidad de comunicar información que posean del estado de la partida, por ejemplo avistamientos de adversarios o aliados, localizaciones de zonas ventajosas estrategicamente en el mapa, etc, que hayan descubierto o que otros agentes les hayan transmitido.

En conjunción con la característica anterior, este aspecto facilita y promueve la aparición de comportamientos emergentes al nivel de los equipos, y esta característica no es compartida por ninguno de los juegos de competición expuestos en el estado del arte, pues en StarCraft, aunque los bandos están integrados por muchas unidades, todas son controladas por un único agente que recibe información de todas sus unidades y las controla a todas.

3.2 Generador de mapas

Con la intención de minimizar el tamaño en memoria de los mapas y poder mantenerlos cargados en su totalidad durante las partidas, se ha optado por una estética minimalista en el desarrollo de los mapas, basada en formas geométricas simples que permitiese una baja densidad de polígonos, y sin texturas.

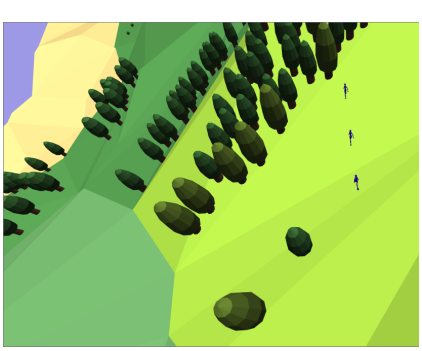
En este trabajo, la generación de mapas se divide en dos partes: la generación de superficies [8] y la generación procedimental de objetos del terreno [4].

La generación de superficies se divide en seis partes:

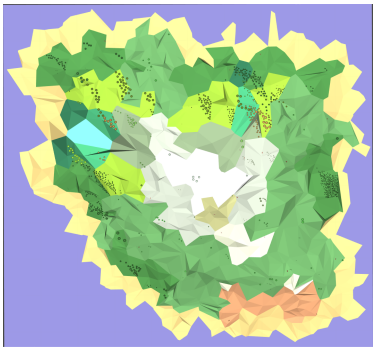
- *Generación de diagramas de Voronoi*: El primer paso de la generación de superficie consiste en realizar una partición en polígonos convexos de un plano rectangular. Para ello, se utilizó el algoritmo de Voronoi.
- *Definición de tierra-agua*: Dado que los mapas que se estan generando son islas, estas se definen dividiendo el plano en sectores de un cierto ángulo respecto al centro del rectángulo (definido como el punto de corte de las dos diagonales) y utilizando ruido rosa para determinar un cierto radio pseudoaleatorio de la isla en cada sector. De esta forma, todos los vértices de los polígonos de Voronoi que caigan dentro del radio de su sector se marcan como tierra y los demás como agua.
- *Definición de la elevación*: La elevación de cada vértice de tierra de cada polígono se define como un factor de su distancia a la costa.
- *Definición de ríos*: Se seleccionan vértices aleatorios usando ruido blanco a partir de una cierta elevación mínima como los orígenes de los ríos. A partir de ahí, se recorre el grafo desde cada origen tomando en las intersecciones la arista de mayor pendiente y aumentando el volumen de agua que lleva el río en ese punto cada vez por cada río que pase por él. Los ríos acaban cuando llegan a la costa o cuando llegan a un vértice degenerado por la redistribución de elevaciones en el que no exista una pendiente máxima bien definida.
- *Definición de humedad*: En este punto, se define la humedad en cada uno de los vértices propagándola desde las fuentes de agua dulce (ríos y lagos).
- *Definición de biomas*: Se sigue el diagrama de biomas de Whittaker [18].

Por otro lado, la generación de objetos del terreno se divide en dos partes:

- *Construcción de objetos*: Existen tres tipos de objetos de terreno: árboles, arbustos y troncos caídos.



(a) Tres agentes del juego *Paintbol* siguiéndose entre ellos.



(b) Ejemplo de mapa generado en la plataforma, usado para las pruebas de este trabajo.

Fig. 3: Ejemplos de gameplay y mapa de juego en PaintBol

– *Distribución de objetos*: En esta parte se utilizan los biomas definidos en el generador de superficies para definir la probabilidad de que haya un árbol o un arbusto en un determinado punto del mapa. El primer paso es seleccionar un conjunto de puntos aleatoriamente en el mapa, y con una probabilidad definida en función del bioma en el que esté el punto, decidir si en cada uno de los puntos hay un objeto. En el caso de los árboles, además, cada árbol deja un número aleatorio de semillas en un anillo a su alrededor. Las semillas se convierten en árboles con una probabilidad que es función decreciente del radio y proporcional al parámetro de probabilidad de árboles o arbustos del bioma del punto estudiado. Este proceso se repite con las nuevas generaciones de árboles producidas por semillas también.

En la figura 3b se muestra un mapa generado con las distintas características explicadas hasta el momento. El color azul representa el agua, mientras el resto de colores representan los distintos biomas del terreno. Los puntos verdes representan árboles, los grises arbustos y los marrones troncos caídos.

4 Experimentación

En esta parte se explican los procesos experimentales llevados a cabo para la verificación y validación de la plataforma desarrollada y se presentan los resultados obtenidos usando un conjunto de agentes de prueba.

4.1 Diseño de experimentos

Para este trabajo, se desarrollaron tres jugadores inteligentes de prueba, con la intención de demostrar el funcionamiento de la plataforma, que además se usarán para realizar un pequeño estudio de resultados que servirán de punto de

referencia para futuras extensiones y usos de la plataforma. Aunque el concepto de juego está diseñado para que cada agente pueda utilizar varios turnos para tomar una decisión, de momento todas las IAs implementadas son atómicas, es decir, deciden acciones cada turno. Cada jugador tiene 2 turnos cada segundo. A continuación, se explica el funcionamiento de cada uno de los agentes creados:

- *RandomAI*: A modo de referencia básica, se ha implementado un primer jugador que toma decisiones aleatorias. Sin embargo, dada la naturaleza continua del juego, se ha optado por no realizar una implementación estricta de lo que sería un jugador aleatorio. En cambio, se ha implementado un árbol de decisión en cual algunos de los nodos dependen de valores aleatorios.
- *BasicAI*: El segundo jugador que se ha implementado sigue un sistema de reglas básico para tomar decisiones, no usa la comunicación y prácticamente no analiza los ruidos que oye. La Figura 4 muestra su grafo de decisión.

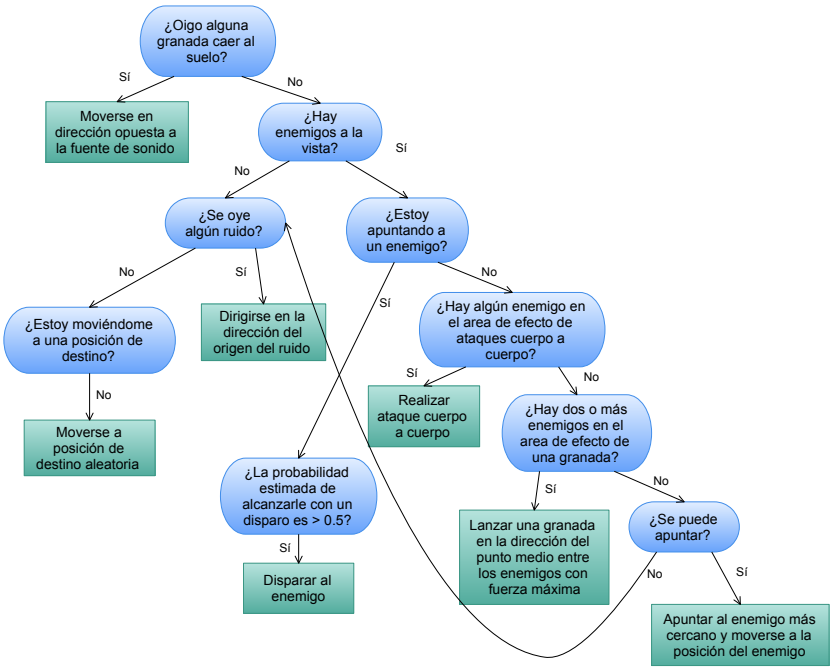


Fig. 4: Grafo de decisión de *BasicAI*

- *HeuristicAI*: El tercer agente implementado hace uso combinado de un árbol de decisión y dos submáquinas de estados para estructurar su comportamiento. La estrategia que sigue es buscar a otros aliados y seguirles. Para poder usar el mismo algoritmo en los cinco jugadores del equipo, los agentes utilizan sus identificadores en el juego para establecer una jerarquía.

De este modo, cada jugador sólo sigue a los jugadores de mayor rango que él. Por otro lado, *HeuristicAI* realiza un mejor análisis de los ruidos que oye el jugador, clasificándolos según unos criterios de prioridad. Además, aprovecha el sistema de comunicaciones para avisar a sus aliados cuando avista a un enemigo, pero sin alertarle.

4.2 Resultados

Table 1: Media y desviación típica de los resultados de las partidas de prueba

MEDIDAS	RandomAI	BasicAI	HeuristicAI
Vidas quitadas	6.90±2.65	18.70±1.71	**25.00±0.00**
Cuerpo a cuerpo	0.10±0.22	0.20±0.27	**3.90±1.53**
Disparando	6.70±2.51	18.50±1.83	**21.10±1.53**
Granadas	**0.10±0.22**	0.00±0.00	0.00±0.00
Con Sigilo	3.50±1.81	14.20±2.83	**15.60±2.65**
A la cabeza	0.00±0.00	0.00±0.00	0.00±0.00
Muertes múltiples	0.00±0.00	0.00±0.00	0.00±0.00
Fuego amigo	0.00±0.00	0.00±0.00	0.00±0.00
Vidas restantes	0.00±0.00	8.60±0.96	**15.80±3.41**
Granadas restantes	9.70±4.66	22.30±1.03	**25.00±0.00**
Puntos por vidas quitadas	10.40±4.30	33.00±4.53	**40.60±2.65**
Bonus por vidas restantes	0.00±0.00	2.70±0.27	**5.80±0.62**
Bonus por granadas restantes	6.60±2.11	**10.00±0.00**	**10.00±0.00**
Bonus por puntería	-2.00±2.56	5.00±2.44	**5.80±2.82**
Bonus por variedad	0.00±0.00	0.00±0.00	0.00±0.00
Puntos totales	*15.00±6.62*	*50.60±5.92*	***62.20±4.56***

En esta sección se presentan los resultados de los enfrentamientos realizados entre los agentes de prueba desarrollados. Se han realizado 5 enfrentamientos para cada posible emparejamiento de los jugadores, en total 15 enfrentamientos, todos en el mismo mapa, del cual se muestra una imagen en la Figura 3b. Los enfrentamientos terminan cuando todos los jugadores de uno de los equipos se quedan sin puntos de vida, y duraron entre media hora y una hora cada uno.

En la Tabla 1 se resumen los resultados obtenidos a través de la media y la desviación típica de cada uno de los parámetros recogidos por la plataforma durante las partidas para cada uno de los jugadores. Los resultados fueron los esperados: los agentes más elaborados obtuvieron consistentemente mejores resultados.

A continuación se procede a realizar un pequeño análisis de algunos aspectos de los resultados. Lo primero que se puede hacer es estudiar de dónde obtiene cada algoritmo la mayor parte de sus puntos, y la respuesta es muy clara: todos los agentes obtienen el grueso de sus puntos directamente de quitar puntos de vida a sus oponentes.

Dado que existen tres tipos de ataque implementados, es interesante observar que la inmensa mayoría de las muertes se obtuvieron mediante disparos, que sólo *HeuristicAI* consiguió una cantidad sustancial de muertes usando ataques cuerpo a cuerpo y que el uso efectivo de las granadas fue despreciable.

Una consecuencia de que ningún jugador usase con éxito las granadas es que varios de los parámetros usados para obtener las puntuaciones de los equipos no han sido aprovechados. Como la única forma de conseguir *muertes múltiples* es usando granadas para alcanzar varios adversarios con un mismo ataque y sólo las granadas pueden herir a los aliados, no se produjeron muertes múltiples ni casos de *fuego amigo*. Por otra parte, tampoco se obtuvo ningún punto por *bonus de variedad*, pues este parámetro recompensaba precisamente el hecho de usar los tres tipos de ataques disponibles. Por otro lado, el reducido uso de las granadas se tradujo en un bonus estable para *BasicAI* y *HeuristicAI* por *granadas restantes*. Tampoco se aprovechó la posibilidad de obtener bonus por *disparos a la cabeza*.

5 Conclusiones y Trabajo Futuro

En este trabajo se ha presentado una plataforma desarrollada sobre Unity3D enfocada en la aplicación y estudio de algoritmos de Inteligencia Artificial en videojuegos. Esta plataforma ofrece características diferentes de las ofrecidas por otras existentes en el ámbito, como la necesidad, por parte de los agentes del juego, de analizar terrenos tridimensionales, crear jerarquías, comunicarse y trabajar en equipo.

La plataforma está compuesta por un juego de estrategia táctica multiagente por equipos en 3D, llamado *Paintbol*, un generador de mapas y una API que permite a los agentes desarrollados por los usuarios interactuar con el juego. Para demostrar el uso de la plataforma, se han implementado algunos agentes de prueba y se han analizado los resultados de los enfrentamientos entre ellos.

El trabajo desarrollado hasta ahora, especialmente en la parte de experimentación con agentes de prueba, sugiere numerosas ampliaciones posibles sobre la plataforma. Entre ellas, cabe destacar la inspección profunda del código de la plataforma y la realización de una buena documentación, de cara a mejorar su extensibilidad, encapsulación, y facilidad de desarrollo con el objetivo de ser publicada en forma de plataforma de competición. Por otro lado, se aplicarán sobre la plataforma múltiples algoritmos de IA, como máquinas de estados, redes neuronales, algoritmos genéticos, etc.

Agradecimientos

Este trabajo está financiado por el Ministerio de Español Economía y Competitividad y los Fondos FEDER Europeos mediante el proyecto EphemeCH (TIN2014-56494-C4-4-P), por la Comunidad Autónoma de Madrid y los Fondos FEDER Europeos bajo el proyecto CIBERDINE S2013/ICE 3095, y por Airbus Defence & Space bajo el proyecto SAVIER (FUAM-076914 y FUAM-076915).

References

1. Alhejali, A.M., Lucas, S.M.: Using genetic programming to evolve heuristics for a Monte Carlo Tree Search Ms Pac-Man agent. In: Computational Intelligence in Games (CIG), 2013 IEEE Conference on. pp. 1–8. IEEE (2013)
2. Berns, A., Gonzalez-Pardo, A., Camacho, D.: Game-like language learning in 3-D virtual environments. Computers & Education 60(1), 210–220 (2013)
3. González-Pardo, A., Palero, F., Camacho, D.: An Empirical Study on Collective Intelligence Algorithms for Video Games Problem-Solving. Computing and Informatics 34(1), 233–253 (2015), http://www.cai.sk/ojs/index.php/cai/article/view/2058
4. Jayelinda: Modelling by numbers. An introduction to procedural geometry. http://jayelinda.com/modelling-by-numbers-part-1a/ (2013)
5. Lucas, S.M.: Ms Pac-Man competition. ACM SIGEVOlution 2(4), 37–38 (2007)
6. Mora, A.M., Montoya, R., Merelo, J.J., Sánchez, P.G., Castillo, P.Á., Laredo, J.L.J., Martínez, A.I., Espacia, A.: Evolving bot AI in Unreal (TM). In: Applications of Evolutionary Computation, pp. 171–180. Springer (2010)
7. Ontañón, S., Synnaeve, G., Uriarte, A., Richoux, F., Churchill, D., Preuss, M.: A survey of real-time strategy game AI research and competition in StarCraft. Computational Intelligence and AI in Games, IEEE Transactions on 5(4), 293–311 (2013)
8. Patel, A.: Polygonal Map Generation for Games. http://www-cs-students.stanford.edu/ amitp/game-programming/polygon-map-generation/ (2010)
9. Robertson, G., Watson, I.: Building behavior trees from observations in real-time strategy games. In: Innovations in Intelligent SysTems and Applications (INISTA), 2015 International Symposium on. pp. 1–7. IEEE (2015)
10. Rodríguez-Fernández, V., Menéndez, H.D., Camacho, D.: Design and development of a lightweight multi-UAV simulator. In: 2nd IEEE International Conference on Cybernetics, CYBCONF 2015, Gdynia, Poland, June 24-26, 2015. pp. 255–260 (2015), http://dx.doi.org/10.1109/CYBConf.2015.7175942
11. Rodríguez-Fernández, V., Ramírez-Atencia, C., Camacho, D.: A multi-UAV Mission Planning videogame-based framework for player analysis. In: IEEE Congress on Evolutionary Computation, CEC 2015, Sendai, Japan, May 25-28, 2015. pp. 1490–1497 (2015), http://dx.doi.org/10.1109/CEC.2015.7257064
12. Rohlfshagen, P., Lucas, S.M.: Ms Pac-Man versus ghost team CEC 2011 competition. In: Evolutionary Computation (CEC), 2011 IEEE Congress on. pp. 70–77. IEEE (2011)
13. Rosca, J.P.: Generality versus size in genetic programming. In: Proceedings of the 1st annual conference on genetic programming. pp. 381–387. MIT Press (1996)
14. Sturtevant, N.R.: Monte Carlo Tree Search and Related Algorithms for Games. Game AI Pro 2: Collected Wisdom of Game AI Professionals p. 265 (2015)
15. Svensson, M.: Dynamic Strategy in Real-Time Strategy Games: with the use of finite-state machines (2015)
16. Togelius, J., Karakovskiy, S., Baumgarten, R.: The 2009 Mario AI competition. In: Evolutionary Computation (CEC), 2010 IEEE Congress on. pp. 1–8. IEEE (2010)
17. Togelius, J., Shaker, N., Karakovskiy, S., Yannakakis, G.N.: The Mario AI championship 2009-2012. AI Magazine 34(3), 89–92 (2013)
18. Whittaker, R.: Whittaker Biome Diagram. http://www.marietta.edu/ biol/biomes/biome_main.htm (1975)