Un videojuego educativo basado en la optimización con colonia de hormigas

Abraham Sánchez L., Martin Garcia M., Miguel A. Jara M., José L. Estrada M.

Benemérita Universidad Autónoma de Puebla, Puebla, México

asanchez@cs.buap.mx, yugiharry@me.com, mikejm92@gmail.com, dirus.umbratrox@gmail.com

Resumen. Tradicionalmente, los videojuegos han provisto un marco de estudio para la inteligencia artificial. El principal objetivo de esta investigación es adaptar el algoritmo de la colonia de hormigas, este se basa en el comportamiento natural de las hormigas para buscar una solución a los problemas que requieren adaptabilidad y trabajo cooperativo. Utilizamos las ideas del algoritmo tradicional de optimización con colonia de hormigas en un agente competitivo dentro de un ambiente 3D, implementado en un videojuego de tipo educativo. Las características más importantes del trabajo son la optimización de cálculos para mejorar la fluidez del videojuego y una arquitectura que permite al videojuego servir como educador para aquellos usuarios interesados en conocer los aspectos generales de esta metaheurística.

Palabras clave: Videojuego, colonias de hormigas, optimización, metaheurística.

An Educational Video Game Based on the Ant Colony Optimization

Abstract. Tradicionalmente, los videojuegos han provisto un marco de estudio para la inteligencia artificial. El principal objetivo de esta investigación es adaptar el algoritmo de la colonia de hormigas, este se basa en el comportamiento natural de las hormigas para buscar una solución a los problemas que requieren adaptabilidad y trabajo cooperativo. Utilizamos las ideas del algoritmo tradicional de optimización con colonia de hormigas en un agente competitivo dentro de un ambiente 3D, implementado en un videojuego de tipo educativo. Las características más importantes del trabajo son la optimización de cálculos para mejorar la fluidez del videojuego y una arquitectura que permite al videojuego servir como educador para aquellos usuarios interesados en conocer los aspectos generales de esta metaheurística.

Keywords: Videogame, ant colony, optimization, metaheuristic.

1. Introducción

El proyecto presentado en este trabajo, está inspirado en el algoritmo ACO (del inglés, Ant Colony Optimization) e implementado en un videojuego. El proyecto pretende demostrar la compatibilidad y utilidad entre la inteligencia artificial y los videojuegos [5], así como optimizar el algoritmo mencionado a través de su implementación en un ambiente 3D y la optimización de los cálculos para evitar que el videojuego pierda fluidez [4]. Después de revisar la literatura, es claro que se trata de una idea innovadora, y no se encontraron trabajos relacionados a videojuegos educativos.

En esta propuesta, el videojuego permite al usuario experimentar los cambios del algoritmo dadas ciertas modificaciones en los parámetros y variables utilizadas por el mismo, además de proporcionar al jugador un panorama similar al de las hormigas, a través de un sistema de penumbra que permite al usuario experimentar la sensación de ceguera con la ausencia de la feromona.

La mecánica de la aplicación tiene como objetivo crear un videojuego de estrategia donde el jugador deberá tomar las decisiones correctas para que un imperio de robots prospere. Para cumplir con este objetivo, el usuario debe administrar los recursos que obtengan sus agentes inteligentes (los robots recolectores) para poder optimizar el algoritmo (modificando los parámetros del mismo a través de edificios). Cada optimización tiene un cierto costo de material/alimento, y cada agente consume una cierta cantidad de alimento conforme avanza el tiempo. El algoritmo que se presenta para cada uno de los agentes antes mencionados, es una mejora al algoritmo de optimización con colonia de hormigas; así como una adaptación del mismo a un ambiente tridimensional, donde el tiempo es un elemento muy importante en la toma de decisiones de los robots.

Además, el jugador tendraá una dependencia con la feromona para poder visualizar el mapa; ya que la iluminación del videojuego está basada en la cantidad de feromona contenida en cada cuadrante. Gracias a todos estos elementos, el jugador será capaz de reconocer los efectos que cada parámetro del algoritmo provoca en la recolección de alimentos de los robots.

La aplicación fue implementada utilizando la "plataforma de desarrollo para creación de videojuegos y experiencias interactivas 3D y 2D": Unity 3D en su versión 5 [1], la cual permitió crear el escenario 3D, agregar física a los cuerpos del videojuego, calcular las distancias y tiempos en el ambiente en tiempo real, así como otras funciones secundarias como aumentar la calidad gráfica del juego, etc. El lenguaje de programación utilizado fue C# y las distintas funciones de la aplicación han sido repartidas en múltiples scripts que permiten a todos los elementos funcionar de manera independiente.

En la seccin dos, se presentan algunos aspectos relacionados con el desarrollo de esta propuesta de trabajo. El modelo matemático del algoritmo de la colonia de hormigas y su adaptacin a la propuesta de este trabajo, se presentan en la sección tres. La sección cuatro detallas los elementos más importantes en el desarrollo del videojuegob y finalmente la sección cinco resalta las conclusiones.

2. Desarrollo del algoritmo

El algoritmo de la optimización con colonia de hormigas se inspira en el comportamiento natural de las hormigas en su entorno natural. Las hormigas deambulan libremente dejando una diminuta cantidad de feromona (de forma aleatoria), hasta encontrar algún alimento. Una vez que las hormigas encuentran el alimento, el rastro de feromona se vuelve más intenso; si antes liberaban una pequeña cantidad de feromona, ahora liberan una cantidad constante de feromona. Este comportamiento permite a las hormigas comunicarse entre ellas y seguir rastros de feromona que de alguna forma marcan el camino indicado. Cuando las hormigas que están en un estado de libre recorrido, detectan un rastro de feromona, éstas alteran su comportamiento y comienzan a avanzar guiadas por la feromona; hasta que eventualmente alcanzan el alimento que alguna hormiga probablemente encontró poco tiempo antes.

Después de un cierto tiempo, el alimento se termina, pero las hormigas continúan siguiendo el camino marcado por la feromona. Al desaparecer el alimento, las hormigas que siguen el rastro de feromona rebasan el límite del camino marcado y poco a poco forman su propio camino sin feromona (sin un alimento que frene su avance). Esto provoca que la feromona se evapore lentamente, haciendo que ese camino que ya no conduce a ningún alimento sea menos elegido [2].

Entre más corto sea el camino recorrido por las hormigas, más se concentrará la feromona debido a que las hormigas recorren constantemente un mismo camino y tardan menos tiempo en volver a pasar por el mismo punto, evitando la evaporación de la feromona y fomentando la concurrencia de las hormigas en ese camino. El algoritmo es implementado con un grupo colectivo de agentes inteligentes, que simula el comportamiento de las hormigas, recorriendo un grafo que representa un cierto problema a resolver utilizando mecanismos de cooperación y adaptación [2], [3].

En la implementación de de esta propuesta de trabajo, se sustituyeron las hormigas por robots, representando el problema de forma que estos puedan actualizar sus soluciones, en base a reglas de probabilidad sustentadas en los niveles de feromona de cada camino posible. Además, es importante plantear una heurística σ que mida la calidad de cada dirección que se tomará. Analizando los posibles nodos que podría elegir un robot y seleccionando el más óptimo según la heurística elegida. Dado que la selección de la dirección a tomar está sujeta en todo momento a funciones probabilísticas; la heurística permite definir el camino más probable a seguir, es decir, la selección de caminos es relativamente aleatoria. Finalmente, es necesario definir una regla τ que permita actualizar la feromona.

3. Modelo matemático del algoritmo propuesto

El modelo matemático del algoritmo constituye su elemento más importante. Este se encarga de controlar las iteraciones para realizar las actualizaciones de

Fig. 1. Robot o agente inteligente propuesto para la implementación del algoritmo

los nodos y sus feromonas, lo que permite a los robots elegir la dirección que tomarán. De igual manera, el modelo matemático aporta al jugador la opción de modificar la inteligencia de cada uno de los agentes para encontrar alimento y regresar a la base (a través de sus parámetros). Tradicionalmente, el algoritmo está enfocado en buscar una solución a un cierto problema. Un ejemplo de un problema común es el agente viajero, en donde se busca encontrar el camino más corto de una ciudad a otra a través de un grafo. Para este proyecto, los robots están programados para buscar comida (materiales que les sirven de alimento).

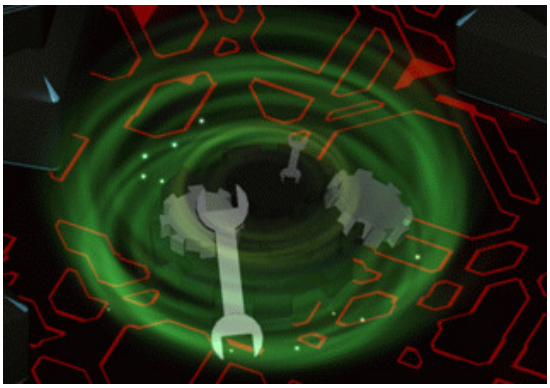

Fig. 2. Alimento que deben encontrar los robots

En el algoritmo tradicional [2], los agentes recorren un grafo donde en cada "turno", cambian de nodo; esto quiere decir que en el algoritmo tradicional la medida de tiempo está basada en turnos y no en mediciones reales de tiempo. La nueva representación del tiempo en el modelo matemático del algoritmo fue la primera aportación de esta investigación; es decir, las actualizaciones se realizan

en tiempo real. Esto significa que los robots están en constante movimiento y no están sometidos a "turnos", sino más bien a cambios de estado. Se propuso utilizar cambios de posición en relación a distancias en metros y poder medir estos cambios de estado (mediante un sí y un no); esto quiere decir que en lugar de un grafo, los robots están orientados por una matriz y en esta matriz cada cuadro está separado por un metro de distancia de su centro al centro del siguiente cuadro. Estos cuadros son coordenadas en x, z dentro de un mundo 3D. Cuando un robot se aproxima a un punto medio de algún cuadrado de la cuadrícula (por ejemplo: (3,4, 1) se aproxima a $(3, 1)$ si el robot que estaba en esa posición avanza a (3,6, 1) el robot se considerará en el cuadro $(4, 1)$ por aproximación), se considera que se ha presentado un cambio de estado y en ese momento se recalcula la fórmula para determinar hasta este punto, que dirección debe tomar el robot(Norte, Noreste, Este, Sureste, Sur, Suroeste, Oeste, Noroeste).

Se examina a continuación la fórmula tradicional para seleccionar el siguiente nodo a visitar:

$$P_{i,j}^{k}(t) = \frac{[\tau_{i,j}(t)]^{\alpha}[\eta_{i,j}(t)]^{\beta}}{\sum_{t \in N_i^k}[\tau_{i,j}(t)]^{\alpha}[\eta_{i,j}(t)]^{\beta}} j \in N_i^k, \tag{1}$$

donde:

- $P_{i,j}^{k}(t)$ la probabilidad de visitar un nodo.
- N_i^k son los nodos no visitados por el robot.
- τ es el momento (en tiempo real) en que se analiza el nodo.
- i coordenada x.
- j coordenada y.
- $\tau_{i,j}(t)$ es el valor de feromona en el punto i, j en un momento t.
- $\eta_{i,j}(t)$ es la heurística a priori que contiene el nodo i, j en un momento t.
- α peso de la feromona, debe ser mayor o igual a 0.
- β peso de la información heurística.

En la investigación presentada, se determinó que la mejor heurística a priori η para el videojuego debía ser la distancia del nodo con respecto al origen, que es la base (donde los robots depositan la comida una vez que la han encontrado). Gracias a esta heurística, los robots pueden encontrar los alimentos más cercanos a la base y, una vez que han obtenido alimento, determinar el camino más corto de regreso a la colonia (a veces ignorando el camino que recorrieron para llegar hasta ese nodo). En este trabajo, el algoritmo original de optimización por colonia de hormigas fue modificado en distintos aspectos; para lograr una adaptación que aumentara la inteligencia de los robots y así mejorar el entretenimiento que el juego aporta. Una de las mejoras más importantes que se debe mencionar con respecto a la fórmula tradicional antes presentada, es que los robots tienen preferencia por ciertas direcciones así como distintos tipos de feromona.

La fórmula $[\tau_{i,j}(t)]^{\alpha}[\eta_{i,j}(t)]^{\beta}$ se modifica de la siguiente manera durante la selección del siguiente nodo a recorrer:

1. Si el nodo analizado se encuentra hacia la misma dirección que el robot recorría, la fórmula cambia a:

$$[\tau_{i,j}(t)]^{\alpha}[\eta_{i,j}(t)]^{\beta} \times 2. \tag{2}$$

2. Si el nodo analizado contiene un camino exitoso que debe recorrer, es decir, se le agrega prioridad a los nodos por donde haya caminado un robot que lleve alimento, la fórmula es:

$$[\tau_{i,j}(t)]^{\alpha}[\eta_{i,j}(t)]^{\beta} \times 15. \tag{3}$$

3. Si el robot carga alimento, la heurística es potenciada y el robot aumenta su preferencia por los nodos cercanos a la colonia, conforme a su valor de inteligencia I (existen edificios que pueden mejorar la inteligencia de los robots):

$$[\tau_{i,j}(t)]^{\alpha}[\eta_{i,j}(t)]^{\beta+I}. \tag{4}$$

4. Finalmente, para seleccionar la siguiente dirección que debe tomar el robot, se realiza un sorteo aleatorio donde la probabilidad (obtenida por la fórmula anterior) de ser un nodo útil, aumenta la posibilidad de tomar determinada dirección. Este sorteo se realiza únicamente entre tres nodos distintos (esto reduce el análisis de nodos y además permite simular de una forma más realista el recorrido de los robots). Por ejemplo: Si un robot camina hacia el norte, los nodos a analizar serán el Noreste, Norte, y Noroeste, por ser los más cercanos al norte.

Los datos anteriores dependen de un valor de feromona τ que se actualiza constantemente. Para obtener este valor de feromona en cada nodo, el método tradicional utiliza la siguiente fórmula y la asigna a cada nodo en cada "turno":

$$\tau_{i,j}(t) = (1-\rho) * \tau_{i,j} + \sum_{k} \triangle {\tau_{i,j}}^{k}, \tag{5}$$

donde:

- ρ es el rango de evaporación de la feromona $0 < \rho < 1$.
- $\sum_k \triangle {\tau_{i,j}}^{k}$ es el cambio de feromona en el nodo y se calcula con la fórmula:

$$\frac{Q}{L_k(t)}, \tag{6}$$

  donde Q es la constante de actualización de la feromona y $L_k(t)$ es el costo en longitud del nodo hacia la base.

Sin embargo, en un ambiente como el de un videojuego donde las operaciones se deben realizar constantemente; calcular todos los valores de feromona en cada uno de los nodos del nivel, es muy costoso en recursos computacionales. Es por ello que en esta propuesta, se implementa una solución que consiste en calcular el valor de la feromona en el nodo, solo en el momento en que se pretende analizar si el nodo será ocupado o no (es decir, cuando los robots cambien de posición). De esta manera, cada uno de los nodos cuenta con una variable personal que almacena el último momento en que fue evaluada la feromona del nodo, y cuando el robot requiere utilizar el nodo, se calcula el tiempo que ha transcurrido desde

esa actualización, y para simular la evaporación de la feromona, se le somete a una función exponencial expresada con la siguiente fórmula:

$$(1-\rho)^{\text{tiempo actual}-\text{ultima actualización}} * \tau_{i,j} + \frac{Q}{L_k(t)}. \tag{7}$$

Así, la variable de actualización se está relacionada con el tiempo, pero se puede posponer su actualización hasta el momento en que se utilizará. A continuación se detalla la implementación de esta propuesta en un videojuego de tipo educativo.

4. Implementación en el videojuego

El videojuego consiste en asegurar la supervivencia de una base de robots. Estas máquinas, interactúan con su medio ambiente a través del algoritmo de optimización por colonia de hormigas modificado. Las máquinas se pueden mejorar buscando algunos edificios que se encuentran en el juego. El jugador debe cuidar su almacenamiento de alimento, así como proteger a los robots de las patrullas enemigas que aparecerán eventualmente. Cada robot evalua la mejora presentada del algoritmo, y los valores de feromona dentro de la matriz de nodos, será un factor crucial en el juego, debido a que el sistema de penumbra impedirá que el jugador pueda ver con claridad, a menos que los robots esparzan su feromona por el mapa, activando las luces dentro del mismo.

4.1. Controles

Debido al objetivo y al gameplay del juego, el videojuego se maneja prácticamente a través de botones. Sin embargo, el jugador puede utilizar distintos controles para interactuar con la aplicación, mejorando la comunicación entre el humano y el software y garantizando que el juego sea más divertido.

- **Cámara**. La cámara del videojuego es totalmente controlable, esta nos permite explorar el mapa sin problemas, moviéndonos hacia arriba, izquierda, abajo y derecha con los botones W, A, S, y D sucesivamente, o bien con los botones de dirección del teclado (botones de flecha). Además, para poder apreciar con detalle los elementos del videojuego, se ha implementado la función de acercamiento y alejamiento de la cámara. Dicha función es manejada con los botones + para acercar la cámara y − para alejar la cámara. A partir de cierta distancia de cercanía, la cámara comenzará a rotar para ofrecer una perspectiva distinta del juego, dicha rotación será nulificada conforme se aleje la cámara de regreso a su posición original o de forma más lejana.
- **Borrado de la feromona**. El botón derecho del ratón permite mostrar y borrar caminos de feromona, posicionando el apuntador sobre una de las esferas azules y manteniendo el botón derecho oprimido.

4.2. Sistema de penumbra

Al inicio de cada ejecución, el mapa está completamente obscuro y el jugador no puede ver más que la base y sus robots. Conforme los robots esparcen feromona por los nodos que recorren, una serie de luces comienzan a encenderse y poco a poco el mapa es más visible. El sistema de penumbra provee al mapa de una luz cada 20 nodos (metros), dentro de cada cuadrante se realiza un promedio de feromona, la cual decidirá la intensidad con la que se enciende cada luz. Si la feromona corresponde a un camino de regreso de los robots (las esferas azules), la iluminación del cuadrante será mucho más rápida, ver la siguiente figura.

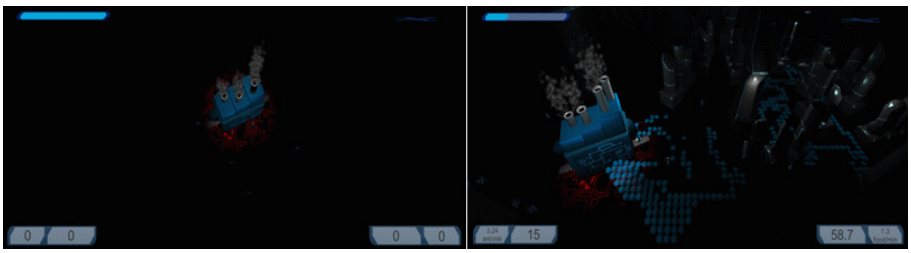

Fig. 3. Al inicio del juego, el jugador se encuentra en penumbras y conforme avanza en el juego, las luces adoptan la forma de la feromona

4.3. Enemigos

Los robots se enfrentan a diversos peligros como la inanición (cuando no hay alimento, los robots comienzan a morir); una esquina donde se encuentra una patrulla que destruye a los robots; y conforme la base crezca y el número de robots aumente, habrá una invasión de pequeñas patrullas que perseguirán a los robots por todo el mapa.

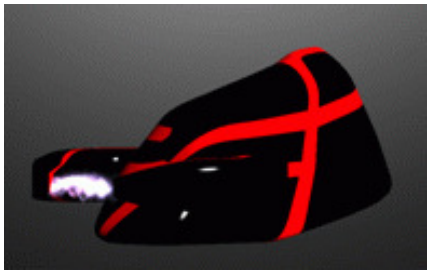

Fig. 4. La patrulla persigue y destruye a los robots

4.4. Edificios

Para realizar mejoras a los robots, el jugador tiene la oportunidad de comprar edificios pagando una cierta cantidad de comida. Una vez construidos, estos e-dificios aportan ciertas ventajas al juego, como protección o perfeccionamiento de la inteligencia y la capacidad de supervivencia de los robots.

En total, el juego cuenta con cuatro diferentes edificios, cada uno con su habilidad y apariencia propia. Uno de estos cuatro edificios (la colonia) se construye cada vez que se inicia el juego, por lo tanto el jugador no debe gastar comida para construirlo. Los cuatro edificios son:

- **Base**: La base es la piedra angular del videojuego, sin ella no sería posible el funcionamiento del videojuego. La base realiza el mapeo del escenario en una matriz de $N \times M$ y guarda los valores de la feromona para cada nodo. Además, gracias a la base es posible mostrar la feromona como objetos transparentes de color azul, facilitando el seguimiento y el borrado de la misma por parte del jugador. Otra de las funciones de la base, es la generación de los robots en un cierto tiempo y el cumplimiento de ciertas condiciones (el tiempo de espera, la cantidad de comida necesaria, etc.).
  La base también se encarga de realizar el conteo de robots y de comida que se visualizan en la Interfaz General del Usuario. Solo es posible tener una base en el videojuego.

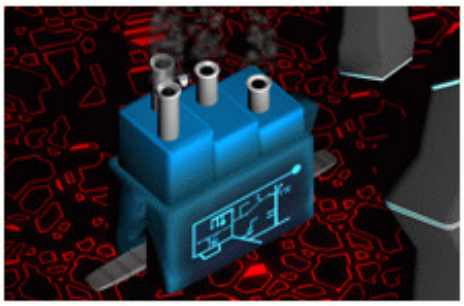

Fig. 5. La base de los robots, donde estos deberán depositar el alimento

- **Torres**: Uno de los mayores retos del videojuego es asegurar la supervivencia de la colonia de los diversos enemigos que rondan a esta. Para protegerse de las patrullas que intentarán destruir a los robots, el jugador puede posicionar torres de defensa que atacarán a las patrullas periódicamente, si estas se acercan lo suficientemente. No existe un límite para colocar torres; sin embargo el jugador debe tener presente que el espacio para colocar edificios es limitado, por lo que deberá colocar todas las torres de una manera estratégica.
- **Máquinas de engranes**: La máquina de engranes es una fábrica que con el paso del tiempo crea engranes. Estos pueden ser utilizados para incrementar

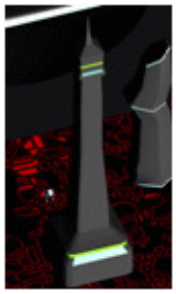

Fig. 6. La torre de defensa permite atacar a las patrullas

la producción de alimento y así favorecer el crecimiento de la base, así como también la recolección de esta materia prima tan importante. Gracias a la máquina de engranes, es posible mejorar la cantidad de alimento aportada por cada engrane; para lograr esto, es necesario hacer clic en el edificio para desplegar el menú de mejoras, y posteriormente utilizar el botón de mejora, siempre y cuando se cuente con la cantidad de alimento requerida para realizar dicha mejora. Cabe mencionar que al igual que las torres, se pueden crear varias instancias de la máquina de engranes si se colocan con cuidado alrededor del mapa.

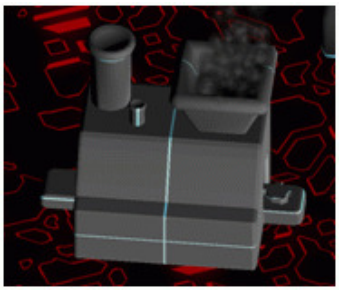

Fig. 7. La máquina de engranes permite crear alimento para los robots

- **Gimnasio**: El gimnasio es un edificio muy útil si se desea optimizar la eficiencia de los robots que están por nacer. Al hacer clic en este edificio, podemos desplegar un menú que ofrece diversas mejoras para los robots. Las mejoras aumentan su precio conforme son compradas, lo que quiere decir que entre más optimizados estén los robots, será más costoso adquirir alimento para la siguiente mejora. Las mejoras que permite realizar el gimnasio son:
  - Duración de la feromona.
  - Velocidad del robot: De 600 inicial hasta 9000 final.
  - Vida del robot.

- Fuerza del robot.
- Inteligencia del robot: De 15 inicial hasta 25 final (sólo afecta a los robots que ya han obtenido alimento y regresan a la base)

En el caso del gimnasio, solo se permite tener una instancia del edificio.

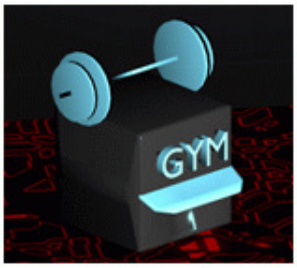

Fig. 8. El gimnasio permite mejorar los parámetros del algoritmo

4.5. Colocación de edificios

Una vez que se ha juntado la cantidad necesaria para crear uno de los edificios disponibles, es posible comenzar la etapa de construcción del edificio. Esta etapa consiste en buscar el lugar adecuado para colocar el edificio. Para evitar abusos en la colocación de edificios, los edificios tienen un límite de distancia entre cada uno, lo cual quiere decir que no es posible colocar dos edificios totalmente juntos o encimados.

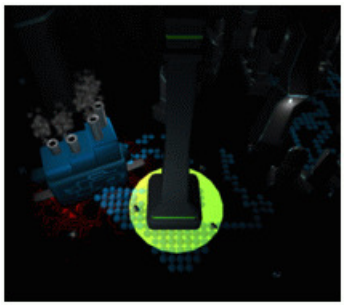

Fig. 9. Una posición válida para colocar un edificio

Durante la etapa de colocación, el usuario deberá mover el ratón a través del escenario; el edificio que se va a construir aparecerá en el mismo lugar en el que se encuentra el apuntador del ratón, así como un objeto que le permitirá saber si la posición es válida o no. Este objeto cambia de color según la proximidad del

futuro edificio con los edificios ya colocados, se pondrá de color rojo si la posición es inválida y de color verde si la posición es válida. Una vez que el usuario esté satisfecho con el lugar de posición, se puede proceder a hacer clic izquierdo para colocar el edificio en su lugar.

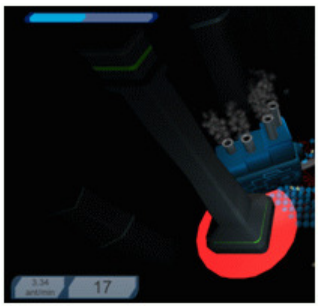

Fig. 10. Una posición inválida para colocar un edificio

La siguiente figura muestra una instantánea del video juego propuesto en este trabajo.

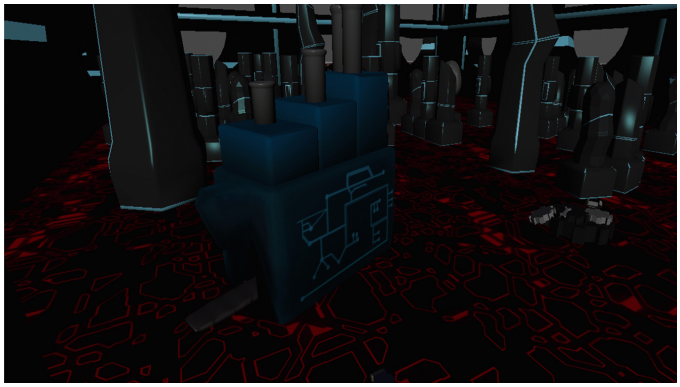

Fig. 11. Ejemplo del video juego propuesto

5. Conclusiones

El algoritmo de optimización con colonia de hormigas tradicional fue adaptado de tal forma, que se puede afirmar que la similitud del comportamiento natural de las hormigas es verdaderamente sorprendente. Estos comportamientos son totalmente plausibles y visualizados en un videojuego 3D. Incluso, el jugador

no es capaz de observar si hay o no feromona en el videojuego. Lo que nos lleva a afirmar que la adaptación se ha llevado de forma satisfactoria al videjuego. Incluso podríamos aventuranos a afirmar que se han hecho ciertas mejoras, obviamente al caso de su adaptación a videojuegos.

Para la realización de este proyecto fue necesario utilizar las tres reglas del algoritmo clásico:

1. Definir correctamente el problema: Conseguir la mayor cantidad de comida posible y regresarla a la base lo antes posible.
2. Plantear la heurística σ: Buscar los nodos más próximos a la base.
3. Definir una regla τ que permita actualizar la feromona: Aportación de la fórmula exponencial para calcular el tiempo transcurrido entre una y otra actualización de la feromona.

Gracias a estos tres puntos, fue más sencillo definir las reglas del videojuego: Conseguir la mayor cantidad de comida en el menor tiempo posible para hacer crecer la base (en número de robots).

El factor tiempo al principio significó una gran limitante, ya que al actua-lizar en todo momento los valores de la feromona, el video juego desarrollado consumía una gran cantidad de recursos de cómputo y era muy complicado utilizar el videojuego (por su relativa lentitud). Más tarde, esta limitante se convirtió en la regla de actualización de la feromona y esto pudo mantenerla dentro de un margen controlado. Se formuló un esquema dentro del cual, el consumo de la feromona depende exponencialmente del tiempo que ha transcurrido desde la última actualización de la feromona (los nodos no son necesariamente actualizados a cada momento).

La combinación de la inteligencia artificial con videojuegos ha demostrado que ambos son completamente compatibles con la observación de que es importante cuidar el número de cálculos realizados al mismo tiempo ya que esto puede retrasar el tiempo de respuesta del videojuego, y aportar una experiencia no deseable para el jugador.

Referencias

1. Unity Technologies: The best development platform for creating games. Obtained from: http://unity3d.com/unity, (2015)
2. M. Dorigo, T. Stützle: Ant colony optimization. Bradford Books, First Edition (2004)
3. E. Martin, M. Martinez, G. Recio, Y. Saez: Pac-mAnt: Optimization based on ant colonies applied to developing an agent for Ms. Pac-Man. In: IEEE Symposium on Computational Intelligence and Games (CIG), pp. 458–464 (2010)
4. G. N. Yannakakis: Game AI revisited. In: Proc. of ACM Computing Frontiers Conference, pp. 285–292 (2012)
5. Z. A. Algfoor, M. S. Sunar, H. Kolivand: A comprehensive study on path-finding techniques for robotics and video games. International Journal of Computer Games Technology, Vol. 2015 (2015)