Simulador de tiro de tanques: NeoNahuel II

Lucas Enrique Guaycochea[1,2], Joel Kaltman[1], María Victoria Galán[1], and Javier Edgardo Luiso[1]

[1] Instituto de Investigaciones Científicas y Técnicas para la Defensa (CITEDEF)
[2] Universidad Nacional de General Sarmiento (UNGS)

Resumen El presente artículo describe el trabajo realizado en el desarrollo de un simulador de entrenamiento para el Ejército Argentino. El simulador NeoNahuel II permite el entrenamiento en el uso del Tanque Argentino Mediano (TAM). Se muestran los análisis y las soluciones a los distintos desafíos tecnológicos que plantearon los requerimientos del sistema solicitado. Se integran conocimientos en computación gráfica, simulación, inteligencia artificial y sistemas distribuidos. Se presentan decisiones de arquitectura y despliegue, y la implementación de algoritmos de inteligencia artificial de los agentes autónomos del simulador. Este simulador constituye una segunda versión desarrollada totalmente in-house por el equipo de desarrollo, a diferencia de su primera versión que utilizaba el juego comercial Steel Beasts [1].

Keywords: simuladores de entrenamiento, computación gráfica, inteligencia artificial, sistemas distribuidos.

1. Introducción

El uso de simuladores para el entrenamiento de procedimientos y operatoria de equipos o máquinas se ha ido expandiendo ampliamente. El concepto de simulación ha ido evolucionando en el tiempo gracias al avance tecnológico logrando simuladores que, a partir de una representación virtual, logran una inmersión tal que el usuario siente estar operando el equipo real en una situación real. La simulación virtual involucra a gente real operando sistemas simulados [2]. Los simuladores logran un beneficio tanto en el ahorro de recursos económicos, como en permitir ejercitar y repetir situaciones, que incluso serían riesgosas de replicar en la realidad. La finalidad última de los simuladores es desarrollar competencias en sus educandos, donde una competencia se define como: "una combinación observable y medible de conocimiento, destrezas, habilidades y atributos personales que contribuyen a la mejora del desempeño del empleado y en el éxito organizacional" [3].

El área de Defensa de varios países ha impulsado el desarrollo en el campo de la simulación al requerir simuladores de entrenamiento para sus fuerzas armadas, buscando el crecimiento de las competencias del personal militar de una manera vasta, económica y medible. La representación virtual de escenarios

logrando imágenes y animaciones (secuencia de imágenes) fotorrealistas contribuyen ampliamente a la inmersión del usuario en la situación de entrenamiento. El avance tecnológico en la computación, más precisamente en la computación gráfica, tanto en el hardware como en los algoritmos de renderizado[3] en tiempo real, viabilizan estos resultados. Otro aspecto fundamental en la fidelidad del entrenamiento es que el usuario interactúe con los mismos elementos físicos que interactuará en la realidad. Para esto, muchos simuladores desarrollan además soluciones de hardware que emulan los dispositivos físicos reales (Fig. 1). Por último, el entrenamiento puede involucrar la interacción del usuario con otros agentes inteligentes en la situación simulada, que pueden ser otros usuarios o agentes simulados. La primera alternativa suele involucrar el desarrollo de un simulador distribuido, donde diferentes usuarios utilizan distintos sistemas interconectados para trabajar sobre un mismo escenario. La segunda requiere el desarrollo de comportamientos realistas para los agentes virtuales que son simulados, este tipo de simulación se la llama constructiva [2] e involucra la implementación de algoritmos de inteligencia artificial.

Figura 1: Izq: Interior de torre del tanque TAM. Der: Cabina del simulador.

El simulador de tiro del Tanque Argentino Mediano, NeoNahuel II, solicitado por el Ejército Argentino, es un producto encuadrado en todo lo descripto en el párrafo anterior. Sus requerimientos han planteado la necesidad de soluciones dentro de las áreas descriptas que fueron totalmente desarrolladas por este equipo de trabajo. Este artículo presenta a continuación una descripción del simulador requerido y llevado a cabo a partir de las soluciones implementadas. Las mismas surgieron luego de un análisis de las necesidades y el estudio de las alternativas vigentes en el cuerpo de conocimientos de las ciencias de la computación y la ingeniería de software. De esta manera primero se describirá la solución a nivel arquitectura y despliegue y luego, a nivel algorítmico y de modelado, se detalla el comportamiento o inteligencia de los enemigos.

[3] proceso por el cual a partir de un modelo virtual de una escena 3D se genera una imagen desde un punto de vista dado

2. Descripción del simulador

El Tanque Argentino Mediano (TAM) es el principal tanque de combate utilizado por el Ejército Argentino. En el ámbito de combate terrestre, el significado que tiene el manejo de estas armas es de gran importancia por su capacidad de influir decisivamente en el desarrollo de las acciones bélicas. La efectividad obtenida no depende de otra cosa más que de la aptitud e idoneidad de su tripulación ante cualquier situación que pueda presentarse. Dentro de este marco, adquiere singular importancia la variedad de entrenamientos que puedan proporcionarse al personal a cargo de su uso, permitiendo un aprendizaje progresivo.

El simulador NeoNahuel II, aquí presentado, es un sistema distribuido que consta de tres programas o aplicaciones. La aplicación Instructor es utilizada para la elección de los diferentes ejercicios, escenarios y cabinas que participarán del mismo. La aplicación Cabina se ejecuta en una cabina física que simula el interior del TAM con los instrumentos de la torre del tanque. Los mandos y controles de la cabina están conectados a una PC que además permite a cada tripulante visualizar el terreno desde su punto de vista mediante los respectivos aparatos de puntería, visores del Jefe de Tanque y Apuntador, respectivamente (Fig. 2). Por último, la aplicación Editor, que permite la creación de ejercicios de forma personalizada para luego poder ser utilizados dentro de la simulación.

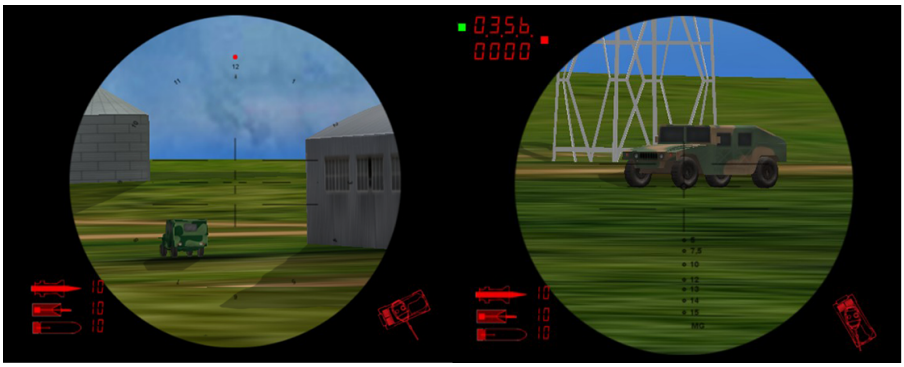


Figura 2: Capturas del sistema. Visualizaciones desde los aparatos de punterías del Jefe de Tanque (izq.) y del Apuntador (der.)

Finalmente, el simulador es una herramienta flexible y económica que complementa el adiestramiento en el terreno del Jefe de Tanque y Apuntador del TAM. Se recrean situaciones de combate que permiten a sus usuarios realizar apreciaciones y mediciones, para resolver en tiempo real situaciones ante un enemigo virtual al que se le podrá disparar y este podrá responder al fuego. Además, permite el entrenamiento a partir situaciones virtuales que combinan todo tipo de dificultades que pueden presentarse en un campo de batalla, desde

inconvenientes técnicos o mecánicos del vehículo hasta la interacción con distintos tipos de enemigos de diferentes embergaduras. Este simulador permite realizar el entrenamiento en forma individual de cada tripulante o de la tripulación completa de un vehículo. También permite realizar ejercicios de duelo de tanques con dos cabinas y adiestrar a una sección completa (3 tanques) en el marco de un ejercicio conjunto.

De esta breve descripción del simulador y los objetivos de instrucción perseguidos, surgen las características y requerimientos técnicos que fueron necesarios implementar para obtener el sistema desarrollado. A continuación se exponen las soluciones desarrolladas que implicaron un mayor trabajo de investigación, estudio, ingeniería e implementación.

3. Despliegue y Arquitectura del simulador

El simulador requiere la participación de varios usuarios con distintos roles de manera simultánea. Esto implica diseñar la solución para ejecutarse en distintas PCs. Se decidió entonces la realización de un despliegue tal que el instructor opere una PC y cada una de las cabinas que representarán un tanque TAM en la simulación utilizarán también una PC. Para cada cabina se desarrollaron los paneles de hardware y la electrónica necesaria para la comunicación de estos con la PC a través de un puerto serie. Se desarrollaron los mandos de ambos tripulantes que luego son vistos por la aplicación como unos joystick estándar. Por último, se conectaron dos monitores que son ubicados en la cabina, tras unas miras con el adecuado lente óptico, donde se muestra la escena virtual 3D en la cual se lleva a cabo el ejercicio.

Las distintas PCs, y las aplicaciones que ejecutan, necesitan estar interconectadas en una red de área local (LAN) para poder comunicarse entre sí. Es una característica del simulador que más de una cabina, y también el instructor, puedan interactuar en una misma realidad simulada, de manera que el modelo de la simulación y la apreciación de la misma a través de su visualización evolucione en forma idéntica para todos los participantes. Para resolver este requerimiento el simulador se construyó sobre la Plataforma de Desarrollos 3D Distribuidos (P3D) desarrollado de manera concurrente por el mismo equipo de trabajo. La P3D es equivalente a un motor de juegos. Brinda particularmente la funcionalidad de un motor gráfico para la generación de las imágenes en tiempo real de la escena virtual e implementa un middleware distribuido que abstrae los aspectos de sincronización de la escena distribuida [5] [6]. En [7] se describen las características de la plataforma P3D y el plan de trabajo actual sobre la misma.

Finalmente, las aplicaciones desarrolladas para este simulador, utilizan como una capa inferior de servicios a la P3D y además crean un protocolo de comunicación entre ellas, implementado mediante TCP/IP. Este protocolo permite administrar la carga de recursos, inicializar y finalizar una escena y un ejercicio, y compartir datos y/o eventos de interés que forman parte de los reportes de cada ejercicio (Fig. 3).

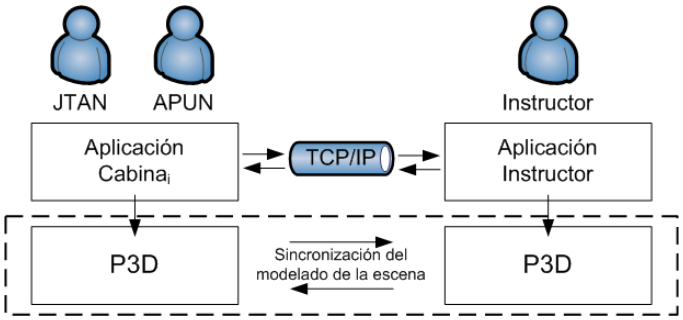


Figura 3: Despliegue y arquitectura del simulador.

4. Comportamiento de Enemigos

Para obtener un simulador con un alto grado de realismo no sólo es importante la calidad de las imágenes fotorrealistas generadas por la plataforma gráfica 3D, sino también la simulación de los comportamientos de los enemigos. La manera en la cual un enemigo se desplaza por el terreno, reacciona ante el contacto con el tanque TAM y procede a atacar, son aspectos que deben ser resueltos y presentarse tan reales como sea posible, siendo esto muy significativo en la experiencia de usuario final. Los algoritmos que son utilizados para resolver estos aspectos pertenecen al área de la inteligencia artificial. En este campo, a las entidades que van a tomar sus propias decisiones se las conoce como agentes autónomos y, en lo que refiere a juegos, como personaje no-jugador o NPC por sus siglas en inglés, *non-playable character*.

El simulador precisa que los vehículos enemigos tomen determinaciones como si estuvieran comandados por humanos. En los ejercicios militares, el terreno donde se desarrolla adquiere un rol fundamental en la toma de decisiones [8]. A nivel requerimientos, el comportamiento esperado de los enemigos es descripto en términos de: "el enemigo se pone a cubierto", "las tripulaciones del TAM deben reaccionar ante una emboscada" o "un vehículo de menor potencia de fuego debe huir a una zona de escondite", entre otros. Además, como se discute en Smelik et al. [8] y en Wells [9], las características de un terreno geo-típico[4] no sólo son esenciales a un nivel táctico y estratégico en un ejercicio, sino que también son fundamentales para los jugadores como claves para orientarse y recorrer el escenario.

La capacidad de un agente autónomo de utilizar las características del terreno como parte de la toma de decisiones o, en definitiva, la relación entre la inteligencia artificial y el terreno, se lo denomina *"razonamiento espacial o del terreno"*[5] [10]. Aparte del modelo o mallado 3d y la textura que son utilizados

[4] En contraposición a un terreno aleatorio cualquiera, es un terreno con características geográficas similares a una zona conocida del planeta

[5] en inglés, spatial/terrain reasoning.

por la plataforma gráfica para la representación visual de la topografía, el simulador precisa disponer de los datos de la superficie y el escenario en un formato conocido y útil a la hora de obtener la información necesaria para los algoritmos de inteligencia artificial. Esto es entonces representado por 3 capas de datos superpuestas de manera tal que puedan ser accedidos eficientemente. Cada capa es una matriz bidimensional que discretiza los valores de la superficie del terreno. La primera capa contiene el valor de elevación, la segunda un valor que identifica el tipo de suelo (pasto, barro, camino, agua, no transitable) y por último la tercera capa que sirve para identificar si esa zona posee algún objeto natural o edificado y la altura del mismo por sobre el nivel de la superficie. Estos datos serán requeridos frecuentemente por los algoritmos descritos a continuación.

4.1. Path Finding

Path finding consiste en encontrar el camino óptimo para llegar a un punto de destino desde una ubicación de origen. A partir de la representación discreta del terreno, se utiliza la variante del algoritmo de camino mínimo de Dijkstra llamado A* *(A-estrella)*, popularmente usado para resolver este problema [11]. Cada celda representa un nodo que esta conectado a sus ocho celdas vecinas. El peso asignado al paso por un nodo esta directamente asociado al valor de tipo de suelo que está en la segunda capa de la representación del terreno; tal que, por ejemplo, el asfalto es asociado con un peso bajo, el barro con uno más alto, y se penaliza las zonas "no transitable" con un peso infinito. Existe un preprocesamiento donde se asigna un valor de "no transitable" en la capa 2 coincidiendo con la descripción de los objetos sobre el terreno de la capa 3. Además, también se tienen en cuenta para la ponderación la pendiente del terreno (para lo cual se consulta la primera capa) y los cambios de dirección.

4.2. Line of sight

Por otro lado se implementó el algoritmo conocido como "línea de vista" (en inglés, *line-of-sight*) que resuelve si hay una visión directa entre dos puntos sin ningún objeto o irregularidad del terreno que se interponga entre ellos. Este algoritmo es utilizado para identificar si en un momento dado un enemigo tiene visualización directa sobre el o los tanques TAM y para encontrar un sector del terreno donde un enemigo puede ocultarse respecto del jugador, esto es, estar a cubierto por alguna elevación o depresión del terreno o detrás de algún objeto.

Para realizar este cómputo el algoritmo hace uso de la información de la primera y tercera capa de los datos del terreno, es decir, la elevación y la existencia de objetos sobre la superficie. Cabe destacar que sólo los objetos de un volumen y opacidad tal que pueden ocultar un vehículo son considerados en esta capa. Por ejemplo, edificaciones como hangares, silos, arboledas son incluidos, y se excluyen objetos tales como rocas, alambrados, arboles poco frondosos y arbustos, entre otros. Similar a lo propuesto en [12], se identifican los dos participantes de esta consulta, el vehículo "observador" y aquel que se quiere definir si es visible por el primero, se toman ambas posiciones sobre la superficie del terreno (junto

con la altura del aparato de visión del primero y la altura media del segundo) y se traza un segmento 3D entre ellos (Fig. 4). Luego, aplicando el algoritmo de Bresenham [13], se discretiza el segmento y se calcula si la altura en cada paso es menor que la suma de las alturas de las capas 1 y capa 3; de ser así, hay un obstáculo entre ellos, y se concluye que la primera posición no puede visualizar a la segunda.

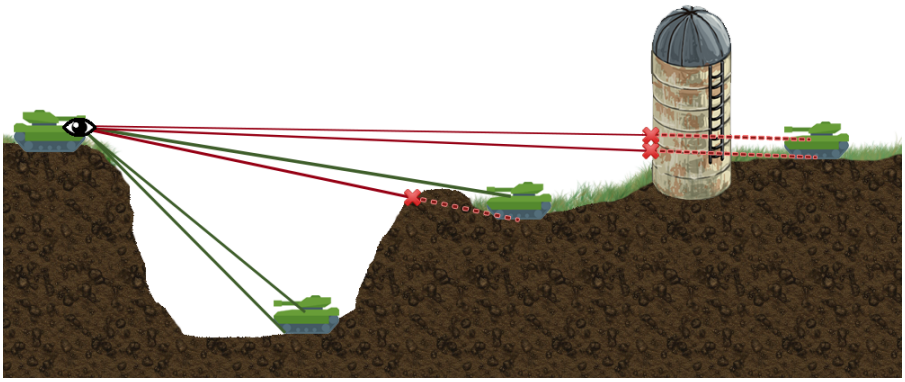


Figura 4: Esquema de las situaciones que resuelve el algoritmo de line-of-sight. Se diferencian las visualizaciones total, parcial y nula de enemigos (de izq. a der.).

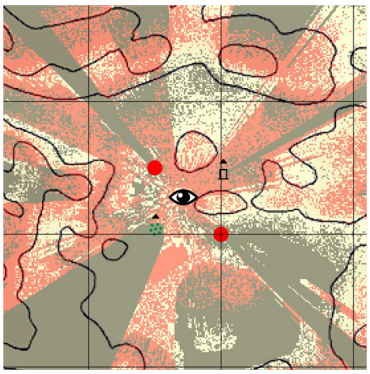
Por último, además de ponerse en práctica el algoritmo durante el transcurso de un ejercicio, se utiliza dentro del editor de situaciones, en donde se permite generar un mapa de visión a partir de una ubicación particular, diferenciando entre posiciones visibles, no visibles y de visibilidad parcial. Las mismas se muestran sombreadas en rosa, en un color claro y oscurecidas, respectivamente. (Fig. 5a).

4.3. Búsqueda de una posición de ocultamiento

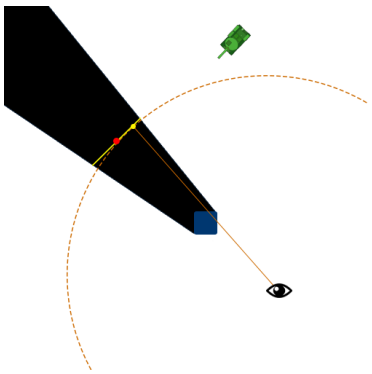
Utilizando el cómputo de la "linea de vista" se desarrolló un algoritmo de búsqueda de posiciones que permite encontrar el lugar óptimo de cobertura en el terreno. Este algoritmo está conformado por una serie de etapas en las cuales se considera si una ubicación y sus alrededores son visibles. Utilizando como base la discretización antes mencionada del terreno, se buscan las posiciones candidatas a ser elegidas como escondite dentro de una circunferencia que tiene como centro la posición del NPC. En caso de no encontrar una opción viable en su interior, se irá ampliando su radio sucesivamente hasta encontrarla. Esto tiene como objetivo evitar excesivas cantidades de procesamiento innecesario en escenarios muy grandes. Entonces se recorren todas las posibles ubicaciones dentro de la

circunferencia, descartando aquellos puntos que estén descubiertos según su linea de visión. Una vez separados los lugares ocultos de los expuestos, se valorizan de acuerdo a dos criterios: distancia al NPC (a menor distancia mejor valor) y visibilidad de posiciones vecinas (si gran cantidad de lugares adyacentes son ocultos, también mejorará su valor). De todos los posibles candidatos se elije el de mejor valoración resultante.

Luego de la implementación del algoritmo descripto, en sucesivas pruebas se encontró que el agente se ubicaba en posiciones donde podía verse parcialmente desde el jugador. Se analizó que este resultado imperfecto es producto de la discretización, ya que el agente, al elegir el punto más cercano de una determinada zona, se ubicaba en un borde de esta y por lo tanto, en ocasiones, no cumplía su objetivo. Es por esto entonces que se debe considerar la totalidad de la zona de ocultamiento que se elige, con el objetivo de acomodarse lo mejor posible dentro de la misma. Para lo cual se plantea una circunferencia con centro en la posición del observador y su radio será la distancia al punto elegido, dejando a dicho punto en el borde de la circunferencia. Se traza una linea tangencial que pase por ese punto y atraviese a todas las posiciones ocultas que rodeen a la elegida. El trazado será detenido en cuanto encuentre puntos visibles para el observador. Finalmente se elige el punto mas próximo al centro de la linea, modificando el punto elegido en caso de ser necesario para "pulir" los resultados imperfectos (Fig. 5b).



(a) Mapa de visión.



(b) Resolución para los casos de ocultamiento parcial.

Figura 5: Resolución del ocultamiento

4.4. Máquina de Estados Finita

Finalmente, queda la tarea de implementar el comportamiento en si mismo, es decir, las distintas actitudes en la que se encuentra un enemigo y la toma de

decisiones que deben realizar. Se definieron 5 comportamientos distintos posibles para estos agentes, estos son, ataque, defensa, huida, marcha pasiva y marcha activa. Por ejemplo, el comportamiento de ataque se describió como: "Cuando el agente *ve*[6] a un tanque TAM decide tomar como objetivo de sus ataques el tanque TAM más próximo a su posición. Tras un tiempo de retardo de inicio de fuego, se fija si se encuentra en rango para realizar un disparo. Si no lo está, decide aproximarse en dirección a su objetivo una determinada distancia y vuelve a fijarse si está en rango. Si se encuentra en rango, abre fuego dos veces y si vence a su objetivo retorna al estado pasivo inicial, sino, decide aproximarse hacia su objetivo para volver a abrir fuego".

Existen distintas alternativas para modelar e implementar un comportamiento de este tipo en la industria, las más conocidas, simples y utilizadas, son a través de un *árbol de decisiones*, *sistemas basados en reglas* y *máquinas de estados finitas* [14]. La primera corresponde a una implementación hardcodeada de cláusulas *if-else*. La segunda consiste en entender el comportamiento como un conjunto de reglas junto con una memoria de datos. Las reglas son representadas por uno o más antecedentes y uno o más consecuentes. Un antecedente es una condición y un consecuente incluye acciones a ejecutar. Existen trabajos que implementan un motor de reglas genérico [15] que permiten definir las reglas a un nivel de abstracción superior que el lenguaje de programación, en un archivo de configuración de texto (esta opción se eligió para la implementación del controlador de cada ejercicio que maneja la instalación de nuevos enemigos, el tiempo de exposición de cada uno y las condiciones de finalización de un ejercicio, entre otras). Por último, un modelo basado en una máquina de estados finitos consiste en identificar los distintos estados en los que se puede encontrar un agente, las acciones que debe realizar en cada uno de ellos y las acciones, condiciones o eventos externos que implicarán un cambio de estado [16].

En este trabajo se eligió utilizar una máquina de estados finita para modelar cada uno de los comportamientos posibles de los enemigos. Esta elección se debió a que presentan una técnica de modelado intuitiva y simple, y requieren una implementación que es sencilla y eficiente.

5. Conclusión

El simulador NeoNahuel II se encuentra desarrollado y está siendo utilizado por el líder de usuario o asesor técnico y sus colegas del arma de caballería del Ejército Argentino en su etapa final de depuración y validación de su funcionamiento. El desarrollo de un producto de estas dimensiones, tras un arduo trabajo de relevamiento y análisis, ha requerido la integración de soluciones a diversos desafíos tanto de hardware como de software, y en distintas disciplinas como computación gráfica, simulación, inteligencia artificial y sistemas distribuidos.

Finalmente se obtuvo un simulador que fue realizado totalmente por este equipo de trabajo, adquiriendo el conocimiento, y logrando un producto com-

[6] Se define que un enemigo *ve a un tanque TAM*, cuando el mismo entra en lo visualizado por alguno de los tripulantes del TAM.

pletamente a medida de lo solicitado por los clientes, pensado desde el inicio para que acompañe el entrenamiento y la formación de los miembros de la caballería, incluso desde antes de su primer contacto con una unidad real del Tanque Argentino Mediano.

Referencias

1. eSimGames: Steel Beasts. 2009, http://www.steelbeasts.com.
2. *Department of Defense Modeling and Simulation (M&S) Glossary*, July 1, 2013. Available at http://www.msco.mil/MSGlossary.html
3. University of Nebraska–Lincoln: The Definition of Competencies and Their Application at NU. University of Nebraska–Lincoln, http://hr.unl.edu/compensation/nuvalues/corecompetencies.shtml, accedido Abril 2017.
4. Ejército Argentino: Tiro con Tanque VC TAM. RFP-02-02, Ejército Argentino, República Argentina, 1995.
5. Guaycochea, L., Luiso, J. y Del Rio Garcia, F.: Middleware P2P para la Sincronización de Eventos Discretos en una Simulación Distribuida de Sistemas que evolucionan en el tiempo. XVIII Congreso Argentino de Ciencias de la Computación, CACIC 2012. Bahía Blanca, Argentina, 2012.
6. Luiso, J., Guaycochea, L. y Abbate, H.: Simuladores de entrenamiento distribuidos: Plataforma de desarrollo para ocultar los aspectos de la distribución. 1er Workshop Argentino sobre VideoJuegos, WAVI 2010, 2010.
7. Guaycochea, L., Luiso, J., Galán, María V. y Abbate, H.: Plataforma de desarrollos 3D Distribuidos (P3D). X Simposio de Informática en el Estado (SIE 2016) - XLV Jornadas Argentinas de Informática e Investigación Operativa (45º JAIIO). Tres de Febrero, Argentina, 2016.
8. Smelik, R.M., Tutenel, T., de Kraker, K.J. y Bidarra, R.: Declarative Terrain Modeling for Military Training Games. International Journal of Computer Games Technology, Hindawi Publishing Corporation, Volume 2010.
9. Wells, W.D.: Generating enhanced natural environments and terrain for interactive combat simulations (GENETICS). en Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST '05), pp. 187-191, ACM Press, New York, NY, USA, 2005.
10. van der Sterren, W.: Terrain Reasoning for 3d Action Games. En Game Programming Gems 2, Ed. Cengage Learning, 2001.
11. Cui, X. y Shi, H.: A*-based Pathfinding in Modern Computer Games. IJCSNS International Journal of Computer Science and Network Security, VOL.11 No.1, Enero 2011.
12. Seixas, R.B., Mediano, M., Gattass, M.: Efficient Line-of-Sight Algorithms for Real Terrain Data. III Operational Research Symposium and IV Logistic Navy Symposium - SPOLM'99, 1999.
13. Bresenham, J.E.: Algorithm for computer control of a digital plotter. IBM Systems Journal (Volume: 4, Issue: 1), 1965.
14. Millington, I. y Funge, J.: Artificial Intelligence for Games. 2da edición, Ed. Morgan Kaufmann Publishers, 2009.
15. Wright, I.P. y Marshall, J.A.R.: RC++: a rule-based language for game AI. First International Conference on Intelligent Games and Simulation (GAME-ON), 2000.
16. Buckland, M.: Programming Game AI by Example. Ed. Wordware, 2004