DISEÑO Y DESARROLLO DE UN VIDEOJUEGO EDUCATIVO CON TÉCNICAS DE INTELIGENCIA ARTIFICIAL PARA ANDROID APLICANDO LA METODOLOGÍA OOHDM. CASO DE ESTUDIO: LABERINTO EN 3D

Soledad González, Margarita Zambrano, César Villacís, Carlos Prócel, Andrés Bustamante
Departamento de Ciencias de la Computación, Universidad de las Fuerzas Armadas"ESPE", Sangolquí,
Ecuador,
cami_sol48@hotmail.com, mezambrano, cjvillacis, ctprocel, cabustamante{@espe.edu.ec}

RESUMEN

El presente trabajo explica el desarrollo de videojuego educativo en 3D conocido como Laberinto que se lo ha realizado empleando Autodesk Maya para el modelamiento del mundo virtual. Como componente fundamental para el proceso de desarrollo de la aplicación y del avatar del juego, se utilizó el lenguaje de programación C# con MonoDevelop junto con el motor de videojuegos Unity. Incluye el uso del lenguaje de modelamiento unificado UML y la aplicación de la Metodología de Diseño de Hipermedia Orientado a Objetos, para el diseño y desarrollo del videojuego. Finalmente se utilizaron técnicas de Inteligencia Artificial para que la computadora pueda generar los laberintos y el avatar del juego pueda resolverlo. El resultado del proyecto desarrollado constituye una aplicación de software definido como un juego educativo en 3D, para estimular el desarrollo cognitivo de los niños entre 7 y 11 años de escuelas, en lo referente al pensamiento lógico y espacial.

***Palabras Clave**:* Juegos educativos, laberintos en 3D con IA, Aplicaciones 3D de entretenimiento, OOHDM, UML, Modelado de juegos en 3D.

ABSTRACT

The present work explains an educational video game in 3D named as Labyrinth. To carry out this work, we used Maya Autodesk for the modeling of the virtual world. As a fundamental component for the process of development of the application and for the avatar of the game, the programming language C# with MonoDevelop and the game engine tool Unity were used. It includes the unified modeling language UML and the application of the Object Oriented Hypermedia Design Methodology, for the design and development of the video game. Finally, it was used techniques of Artificial Intelligence so that the computer can generate the labyrinths and the avatar of the game can solve them. The result of the developed project constitutes an application of defined software as an educational game in 3D, to stimulate the cognitive development of the children between 7 and 11 years of schools, regarding to the logical and space thought.

***Keywords**: Educational Games, labyrinths in 3D with AI, 3D Applications of entertainment, OOHDM, UML, Modeling of games in 3D.*

1. INTRODUCCIÓN

Los videojuegos constituyen un fenómeno popular que se inserta en el proceso de desarrollo tecnológico que experimenta la sociedad. Los videojuegos se introdujeron por primera vez en los Estados Unidos, a principios de los años setenta con un éxito sin precedentes en los salones recreativos hasta entonces ocupados por máquinas tragamonedas y pinballs (Rodríguez E., 2002). Los videojuegos constituyen una de las actividades de entretenimiento más populares de hoy en día. Además, su campo de actuación, desde la segunda mitad de la década de los ochenta, se ha ampliado y ha sobrepasado la frontera del entretenimiento abriendo posibilidades de uso en el ámbito educativo (Rodríguez E., 2002).

Conviene subrayar que en los últimos años está aumentando sensiblemente la oferta de videojuegos educativos, que se presentan como una alternativa de compra a los videojuegos violentos. Este incremento viene motivado por varios factores, como se extrae de los estudios de (Pérez M., 2008), entre los que destaca la madurez de las empresas desarrolladoras, lo que implica productos de gran calidad con buenos guiones y acabados, entrada en la cadena de distribución, actividades de promoción realizadas con el fin de ser conocidos por el público, la apuesta por el mercado del PC y, en la actualidad, por el de las consolas y, la expansión de las capacidades multijugador de los videojuegos y del hardware.

Para el desarrollo actual de videojuegos se utilizan motores de videojuegos que hacen referencia a una serie de rutinas de programación que permiten el diseño, la creación y la representación de un videojuego. Del mismo modo existen motores

de juegos que operan tanto en consolas de videojuegos como en sistemas operativos. La funcionalidad básica de un motor es proveer al videojuego de un motor de renderizado para los gráficos 2D y 3D, motor físico o detector de colisiones, sonidos, scripting, animación, inteligencia artificial, redes, streaming, administración de memoria y un escenario gráfico. El proceso de desarrollo de un videojuego puede variar notablemente por reutilizar o adaptar un mismo motor de videojuego para crear diferentes juegos (Monedero R., 2013). Para la creación de los escenarios, objetos y animaciones de los videojuegos se utiliza Maya que es un programa de modelado en 3D, creado por la empresa Autodesk. Está orientado a artistas y profesionales del diseño y multimedia. Puede ser usado también para crear, visualizaciones 3D estáticas o vídeos de alta calidad. Además incorpora un motor de 3D en tiempo real el cual permite la creación de contenido tridimensional interactivo que puede ser reproducido de forma independiente (Monedero R., 2013).

La IA es la rama de la ciencia que se encarga del estudio de la inteligencia en elementos artificiales y, desde el punto de vista de la ingeniería, propone la creación de elementos que posean un comportamiento inteligente. Dicho de otra forma, la Inteligencia Artificial (IA) pretende construir sistemas y máquinas que presenten un comportamiento que si fuera llevado a cabo por una persona, se diría que es inteligente. En el caso de los videojuegos la IA hace posible que el usuario juegue contra la computadora, permite controlar las mecánicas de los juegos (Russell Peter, 2010). Esto además se ha podido contrastar en nuestros trabajos recientes como (Villacís, C., et. al, 2014) y (Torres, M. et. al, 2013).

Para efectos de este proyecto, se utilizó el motor de juegos Unity para el desarrollo del juego del laberinto en 3D junto con el lenguaje de programación C#. Para el modelamiento de los escenarios del laberinto y la animación del avatar del juego se utilizó la herramienta Maya. Las técnicas de IA utilizadas fueron algoritmos de planeación para la generación automática de los laberintos y la regla de la mano izquierda para que el avatar resuelva por sí solo el juego. Además se aplicó la metodología OOHDM con UML para el proceso de diseño y desarrollo del videojuego (González S, 2014).

El resto del artículo se ha estructurado de la siguiente manera: En la sección 2, se presenta los métodos empleados que incluyen la información de la metodología OOHDM, UML, así como también el motor de juegos Unity y las técnicas de IA utilizadas en el videojuego. En la sección 3, se puede apreciar el proceso de diseño e implementación. La sección 4, presenta los resultados obtenidos. La sección 5 muestra los trabajos relaciona dos. Finalmente la sección 6 muestra las conclusiones y trabajo futuro.

2. MARCO TEÓRICO REFERENCIAL.

2.1. Metodología OOHDM

Para el desarrollo de esta aplicación multimedial, se adoptó la Metodología de Diseño de Hipermedia Orientado a Objetos (OOHDM), desarrollado por Daniel Schwabe y Gustavo Rossi. Esta metodología consta de cuatro etapas: Diseño Conceptual, Diseño Navegacional, Diseño Abstracto de Interface e Implementación. Cada etapa define un esquema objeto específico, en el que se introducen nuevos elementos o clases (G. Rossi; D. Schwabe; C.J.P. de Lucena; D.D. Cowan, 1995).

En la primera etapa se construye un esquema conceptual representado por los objetos de dominio o clases y las relaciones entre dichos objetos. Se puede usar un modelo de datos semántico estructural, como el modelo de entidades y relaciones. El modelo OOHDM propone como esquema conceptual basado en clases, relaciones y subsistemas.

En la segunda etapa, se define la estructura de navegación a través del híper documento mediante la realización de modelos navegacionales que representen diferentes vistas del esquema conceptual de la fase anterior.

El Diseño Navegacional se expresa, también con un enfoque orientado a objetos, a través de dos tipos de es quemas o modelos: (1) El denominado esquema de clases navegacionales, con las posibles vistas del híper documento a través de unos tipos predefinidos de clases, llamadas navegacionales, como son los "nodos", los "enlaces", y otras clases que representan estructuras o formas alternativas de acceso a los nodos, como los "Índices" y los "recorridos guiados"; y (2) El esquema de contexto navegacional, que permite la estructuración del hiperespacio de navegación en sub espacios para los que se indica la información que será mostrada al usuario y los enlaces que estarán disponibles cuando se acceda a un objeto o nodo en un contexto determina do.

La tercera etapa está dedicada a la especificación de la interfaz abstracta. Así, se define la forma en la cual deben aparecer los contextos navegacionales. También se incluye aquí el modo en que dichos objetos de inter faz activarán la navegación y el resto de funcionalidades de la aplicación, esto es, se describirán los objetos de interfaz y se los asociará con objetos de navegación. La separación entre el diseño navegacional y el diseño de interfaz abstracta permitirá construir diferentes interfaces para el mismo modelo navegacional.

La cuarta etapa, es en si la implementación del híperdocumento o sistema hipermedial diseñado, es decir, la concreción de los modelos navegacionales y de interface en objetos particulares con sus correspondientes con tenidos y sus posibilidades de navegación. Aunque, al utilizar un enfoque de orientación a objetos podría parecer conveniente que la implementación se hiciera en un entorno de construcción de híperdocumentos también orientado a objetos, debido al carácter abstracto del diseño, sin embargo ésta puede hacerse fácilmente en otros entornos hipermediales que permitan trabajar con aplicaciones multimedia. Se cumplirán las siguientes actividades como parte de una metodología de desarrollo estándar: a) Análisis; b) Diseño; c) Desarrollo; d) Pruebas (Pressman R., 2002).

Para el desarrollo de este proyecto se aplicó una investigación bibliográfica de fuentes de Información, y consultas en Internet. Mediante esta etapa del obtuvo la información necesaria y válida que permita establecer el marco teórico referencial, el cual proporcione el soporte teórico – técnico necesario para la consecución del proyecto.

El sistema desarrollado se compone de un componente hipermedial para dispositivos móviles que corre sobre el sistema operativo Android y otro de tipo desktop que corre sobre la plataforma Windows. La Metodología OOHDM se aplicará en conjunto con la Ingeniería de Software y el Lenguaje de Modelamiento Unificado (UML).

2.2. Motor de Juegos Unity

Unity es un motor gráfico 3D para PC y Mac que viene empaquetado como una herramienta para crear juegos, aplicaciones interactivas, visualizaciones y animaciones en 3D en tiempo real. Unity puede publicar contenido para múltiples plataformas como PC, Mac, Nintendo, Wii e iPhone. El motor también puede publicar juegos basados en la web usando el plugin Unity Webplayer, que es un componente que permite interpretar código y los mapas de archivos 3D (Blackman S., 2011).

Entre las principales características de Unity se pueden mencionar: Fácil instalación en ambientes Windows, Mac y LINUX; Interfaz de usuario amigable, compuesta por cinco partes: 1) vista de escena; 2) vista de juego; 3) vista de proyecto; 4) vista de jerarquía; 5) vista de inspector; Modos de visualización 3D en perspectiva; Fácil configuración de la visualización utilizando componentes: 1) Render Mode; 2) Color Modes; 3) Interruptor de luces; 4) Interruptor de skybox, lenseflare y niebla; y Manejo de botones de control para comandos de transformación.

Librerías y Componentes del Unity

Las principales librerías que maneja el Unity son seis como se explican a continuación:

- Creación de escenas: Unity tiene un DLL que maneja la creación de escenas en 3D donde se ubican to dos los elementos u objetos del juego en 3D como planos, edificios, terrenos, cielo, personajes, etc.
- Generador de terrenos: Unity tiene un DLL para generar terrenos los mismos que son generados como una malla plana que se puede texturizar y esculpir sin salir del editor. Los terrenos tienen algunas propiedades importantes, como la longitud del terreno y el nivel de detalle del terreno.
- Renderizado: Unity tiene una DLL para configurar el renderizado de los objetos 3D de un juego y añadir efectos especiales al mismo como: a) Niebla (Fog); b) Color de la Niebla (Fog Color); c) Luz de Ambiente (Ambient Light); d) Material de la Caja del Cielo (Skybox Material); e) Fuerza de la Luz (Halo Strenght); f) Fuerza del Fuego (Flare Strenght); g) Textura de la Luz (Halo Texture); h) Mancha de Galleta (Spot Cookie)
- Controlador de Primera Persona: Unity incluye un DLL para el Controlador Estándar en Primera Persona que hace que los juegos con vista en primera persona sean realmente sencillos de configurar. Para manejar este controlador en la escena, se debe ir a la vista de proyecto y seleccionar "Standard Assets > Prefabs". Dentro de esa carpeta hay un prefabricado (prefab) llamado "First Person Controller". Se arrastra este objeto a la vista de escena y se lo posiciona de forma que el cilindro toque el terreno.
- El componente MonoBehaviour: Utilizado para la programación de scripts de Unity, en dos lenguajes de programación que son el C# y el Java.
- El componente GameApp del Juego: Este componente maneja toda la interfaz de usuario y la computación gráfica del juego y se lo puede integrar a otras clases creadas por el usuario, como en el caso del presente proyecto que consta de más de 20 clases.

3. DISEÑO E IMPLEMENTACIÓN

3.1. Diseño de los Algoritmos con Inteligencia Artificial para Armar el Laberinto

La regla de IA que se adoptó para recorrer automáticamente el laberinto fue el método de la mano derecha, que permite que el jugador controlado por la computadora (JCC) resuelva el laberinto y encuentre la salida, manteniendo la mano derecha en contacto con una de las paredes del laberinto durante todo el recorrido, como se puede ver en la Figura 1. El método se puede aplicar también para la mano izquierda conocido en inglés como "Wall Follower Method". Es tos métodos no garantizan salir pronto del laberinto, ni que el camino seleccionado sea el más corto, no obstante, sí se conoce que se saldrá del laberinto (Loh Peter K. K. and Prakash Edmond C, 2009).

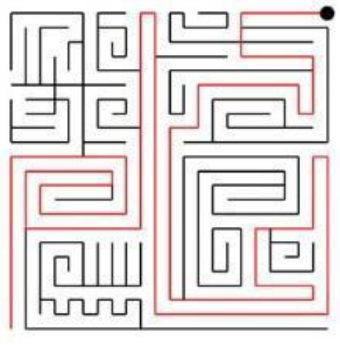

Figura 1. Método de la mano derecha.[1]

3.2. Diagrama de Clases

La Figura 2 muestra las clases que componen el videojuego educativo. La clase ***MonoBehaviour*** proporciona la inicialización básica del modo gráfico de Unity y la lógica del juego. La clase ***GUICamera*** se encarga de controlar la interfaz gráfica del usuario en 2D, donde consta el menú y sus opciones. ***La clase FarCamera*** permite visualizar todo el laberinto a través de una cámara. La clase ***FollowCamera*** permite que una de las cámaras siga al ratón que está en el laberinto.

La clase ***MazeAlgorithm*** se encarga de realizar todas las operaciones matemáticas y generar con IA el mapa del laberinto, lo cual permite obtener una matriz de unos y ceros, donde los ceros representan el camino a seguir y los unos representan las paredes del laberinto. La clase ***MazeCreator*** se encarga de pasar del entorno matemático a un entorno virtual en 3D, para lo cual utiliza la clase ***MazeAlgorithm*** para operar el laberinto. La clase ***MouseBrain*** se encarga de controlar el movimiento del ratón en modo automático, donde se utiliza IA para analizar la ruta a seguir y llegar a la meta que consiste en encontrar el queso y utiliza parte de los algoritmos de la clase ***MazeAlgorithm*** para moverse por el camino representado por un grupo de ceros y almacena en una lista enlazada todas las posiciones recorridas basada en la regla de la mano izquierda. La clase ***Player Controller*** controla manualmente el movimiento del ratón por el laberinto.

La clase ***Loading*** permite manejar la ventana que carga el juego y opera un buffer de comunicación que calcula el porcentaje de carga del mismo. La clase ***GameMode*** controla el tipo de juego que puede ser manual, donde el que opera el juego es un usuario; o también puede ser automático, donde el que opera el juego es la PC con IA. Además esta clase se encarga de almacenar los estados del juego como son el nombre del usuario, el nivel de dificultad (principiante, intermedio, avanzado) y el tiempo que se demora en completar el laberinto. La clase ***ParticleTester*** se encarga de manejar un sistema de partículas para generar los efectos de explosión que tiene el juego y que se produce cuando el usuario representado por un pequeño ratón, encuentra el queso en el laberinto.

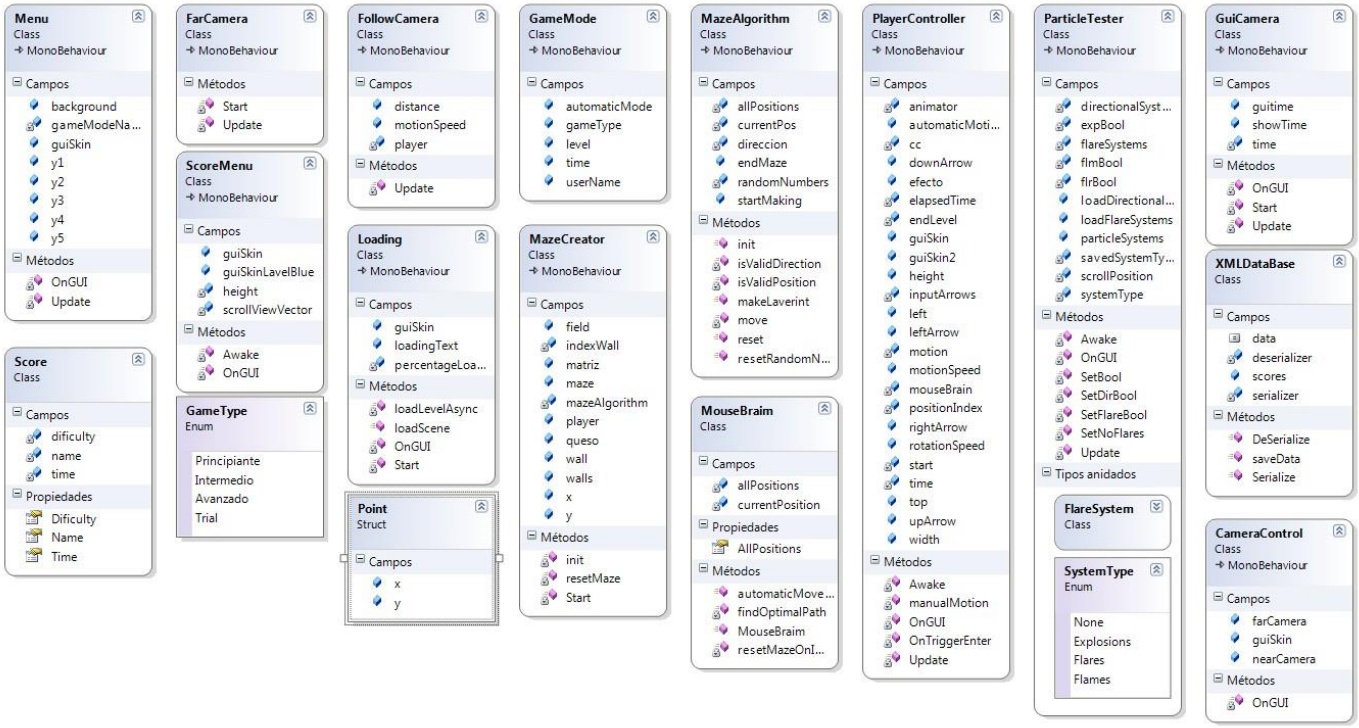

Figura 2: Diagrama de clases del videojuego.

3.3. Diagrama de Despliegue

En la Figura 3 se presenta el diagrama de despliegue del juego, donde se muestran las relaciones físicas de los distintos nodos que lo componen y el reparto de los componentes sobre dichos nodos. Los componentes básicos del juego se cargan en memoria y los matemáticos en el procesador. En cambio los componentes visuales se cargan en la capa de aplicación del sistema operativo que puede ser Windows si la aplicación es desktop; y Android si la aplicación es móvil.

3.4. Aplicación de la Metodología OOHDM

3.4.1. Diseño Conceptual

Durante esta actividad, se construye el esquema conceptual representando clases, los objetos, sus relaciones y las colaboraciones existentes en el dominio del juego. En la Figura 6 se ilustró el diagrama de clases del video juego que se compone por 17 clases, de las cuales las que controlan el modelo de datos de la aplicación son:

a) La clase ***XMLDataBase*** que se utiliza para manejar los archivos planos del juego donde se almacena los datos del mismo; b) La clase ***Scores*** que se utiliza para controlar el puntaje del juego obtenido por los usuarios y la PC; c) ScoreMenu que utiliza la clase Score para determinar el puntaje del usuario de acuerdo al nivel de dificultad del juego; d) La clase ***GameControl*** que controla la ejecución del juego independientemente de las opciones del mismo, como son: 1) Trial; 2) Principiante; 3) Intermedio; 4) Avanzado.

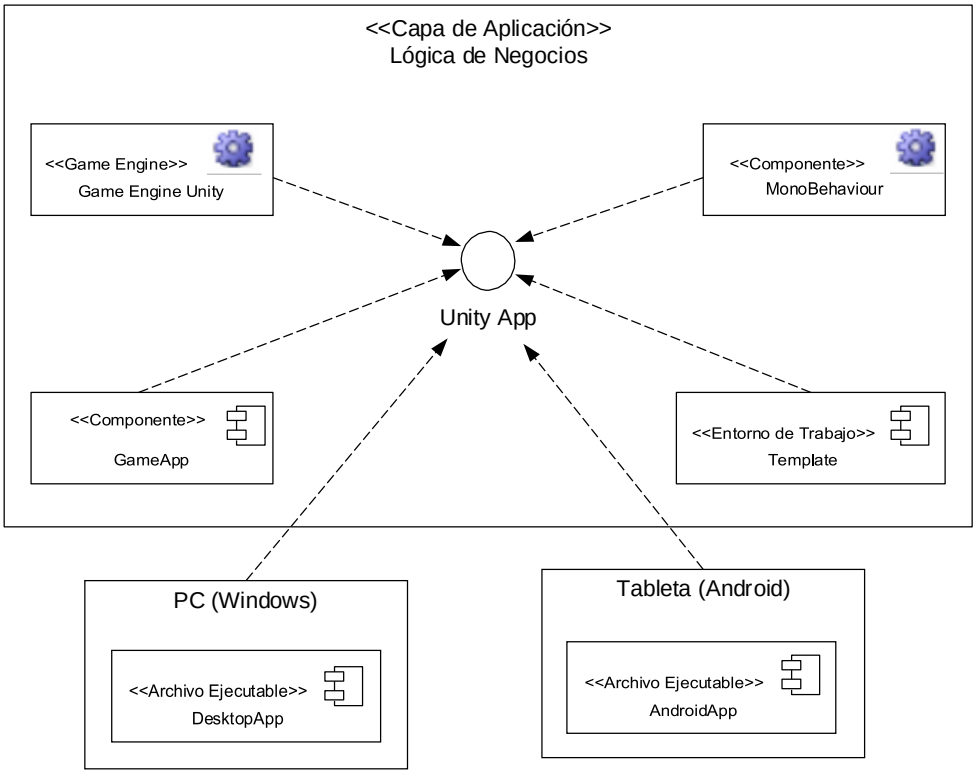

Figura 3: Diagrama de despliegue del videojuego.

3.4.2. Diseño Navegacional

El diseño navegacional se utilizó para construir el modelo de navegación de la aplicación que proporciona un mapa con los nodos y los enlaces del juego, como se puede ver en la Figura 4. El sistema está basado en formularios, por lo tanto contiene una mezcla adecuada de estética, contenido y tecnología. Posee los siguientes objetos y contextos navegacionales:

A) Objetos Navegaciones

- Página del Formulario de Configuración del Juego
- Página del Laberinto de Modo Manual
- Página del Laberinto de Modo Automático
- Página del Formulario de Navegación y Entorno del Juego
- Página del Formulario de los Puntajes del Juego

B) Contextos Navegacionales

- Iniciar Sesión
- Configurar Juego
- Resolver el laberinto de forma manual
- Resolver el laberinto de forma automática
- Guardar Puntajes
- Visualizar Puntajes

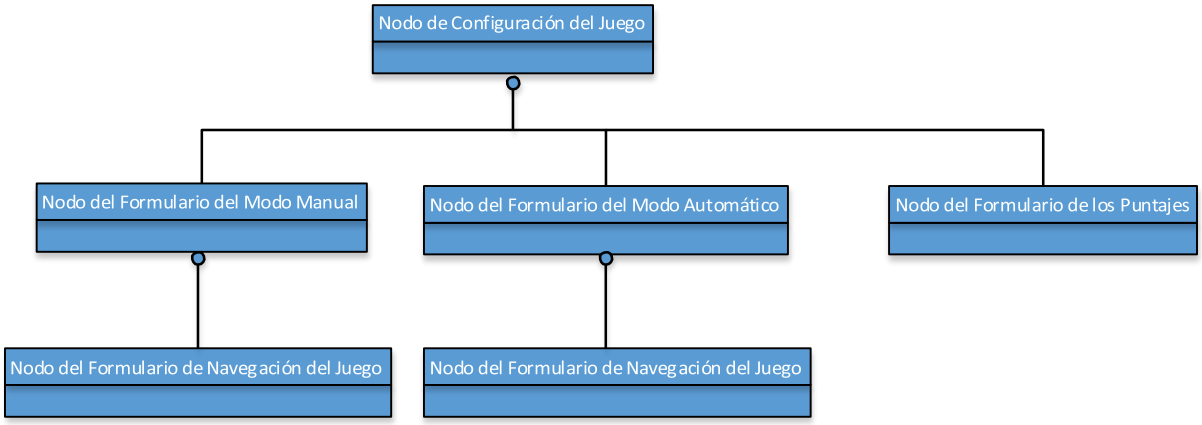

Figura 4: Modelo de navegación del videojuego.

3.4.3. Diseño Abstracto de Interface

El diseño abstracto de interface permitió construir las diferentes interfaces del videojuego par el mismo modelo navegacional. OOHDM utiliza el diseño de vista de datos abstractos (ADVs – Abstract Data View). Estos ADVs tienen la interface y el estado más no la implementación del juego. Las interfaces están basadas tanto en una arquitectura de contenido (i.e. forma en la que los objetos se estructuran para su presentación y navegación) como en una arquitectura de Aplicación Móvil (i.e. forma en la que la aplicación se estructura para gestionar la interacción del usuario). A continuación se detallan las interfaces del usuario Jugador; ya que posee una inter acción directa con el sistema.

- Nodo del Formulario de Configuración del Juego
- Nodo del Formulario del Juego Manual
- Nodo del Formulario del Juego Automático
- Nodo del Formulario de la Navegación y Entorno del Juego
- Nodo del Formulario de los Puntajes del Juego

3.4.4. Implementación

En esta etapa se implementa el diseño como los elementos de información de la navegación, los contextos, las interfaces y la dinámica de la aplicación. Para la construcción del videojuego se utilizó Maya como la herramienta de modelamiento 3D, el motor de juegos Unity para manejar el entorno gráfico de la aplicación y el lenguaje de programación C# de Mono que es un lenguaje multiplataforma.

A) Creación del Modelo 3D en Maya

Maya de Autodesk se utilizó para la creación de los gráficos y objetos, edición de materiales en 3D y la configuración de la iluminación del juego. En la Figura 5 se pueden ver los elementos que se diseñaron en el software Maya para Unity los cuales fueron los siguientes:

- El plano del laberinto.
- Los cubos del laberinto que representan las paredes.
- El ratón que representa al usuario en modo manual y a la PC en el modo automático.

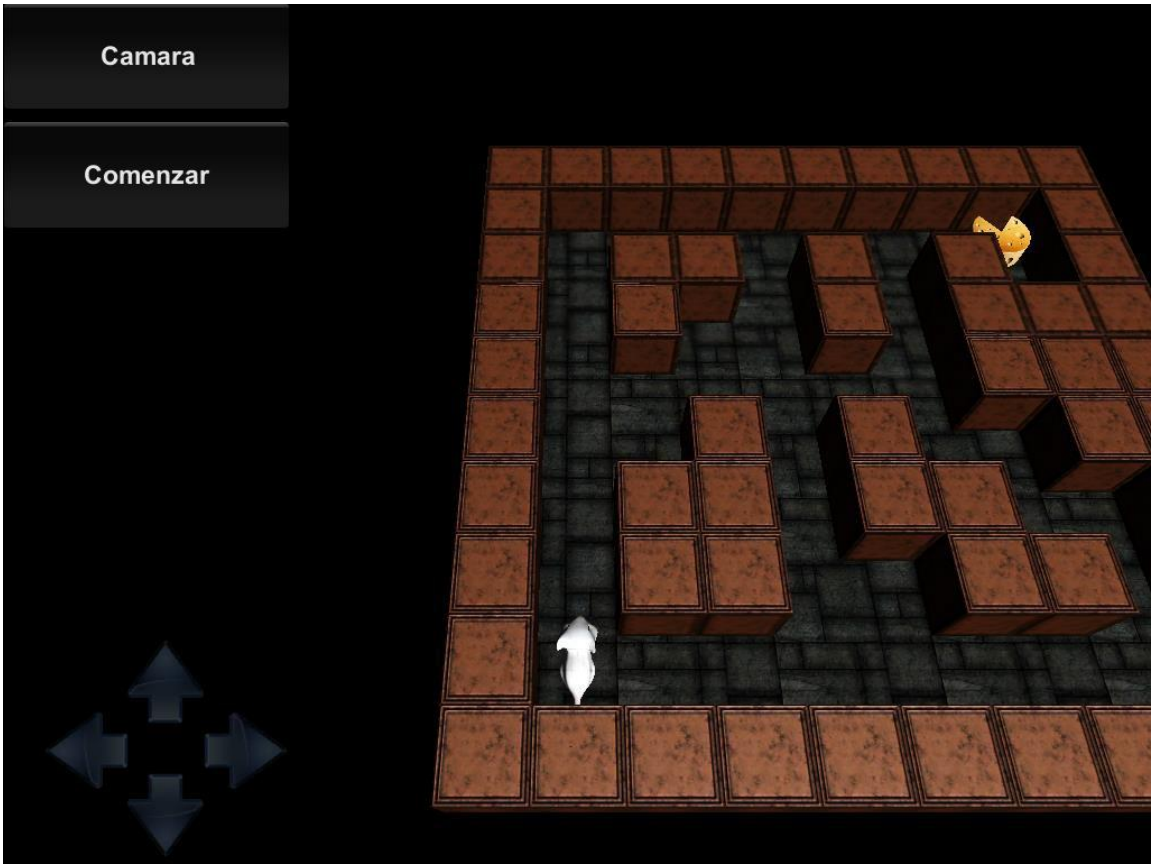

Figura 5. Elementos del Videojuego.

B) Creación del Videojuego con Unity

El videojuego fue desarrollado en su totalidad con el motor de juegos Unity y con la herramienta de programación C# con Mono para la creación de las clases del juego desarrollado. Adicionalmente se utilizó tres componentes básicos del motor de juegos Unity, los mismos que son:

- ***Componente CoreUnity:*** Este componente es el centro de creación de aplicativos del Unity, ya que se constituye en el alma del Game Engine del Unity, donde se encuentran todas las clases que manejan toda la computación gráfica del juego como: vectores, matrices, imágenes, texturas, color, fuentes, objetos 2D y 3D, luces, cámaras, animaciones, física y transformación de objetos.
- ***Componente MonoBehaviour:*** Este componente es un conjunto de DLLs implementados por el proyecto Mono C# para Unity que es compatible con C#.NET y es el que se encarga de compilar el pro grama con las librerías del Unity y manipular los objetos mediante scripts de programación.
- ***Componente GUI:*** Este componente es el que permite crear las interfaces gráficas del usuario en 2D como los componentes para formularios tales como: botones, etiquetas, sliders, etc.

C) Programación del juego con C# de Mono

C# con Mono es una plataforma de software diseñado para permitir a los desarrolladores crear aplicaciones multiplataforma fácilmente. Auspiciado por Xamarin, Mono es una implementación de código libre (open sour ce) del framework .NET de Microsoft basado en las normas de ECMA para C# y el Lenguaje Común de Ejecución (CLR Common Language Runtime). Un conjunto de soluciones crecientes y una comunidad activa que contribuye dinámicamente y de manera entusiasta, está ayudando a que Mono se posicione en el mercado convirtiéndose en la opción principal para el desarrollo de aplicaciones de Linux.

4. EVALUACIÓN DE RESULTADOS

4.1. Evaluación de objetivos alcanzados

Los resultados obtenidos luego del desarrollo del videojuego, con respecto a los objetivos planteados al inicio del proyecto, se pueden apreciar en la Tabla No. 1. De la misma manera la Figura No. 5 muestra uno de los modelos de la aplicación, correspondiente al nivel principiante del juego. Como se puede observar, el modelo 3D del videojuego posee todos los elementos que componen un laberinto como son las paredes, los callejones y el personaje o avatar que representa al usuario en modo manual o a la PC en modo automático según sea el caso. El juego del laberinto en 3D tiene tres niveles de dificultad: a) Principiante; b) Intermedio; c) Avanzado. Además se tiene un modo de entrenamiento que le permite al usuario perfeccionar el dominio de la solución del laberinto en 3D, por lo que se recomienda un estudio que evalúe el impacto en los niños de escuelas entre 7 y 11 años, para medir el nivel de desarrollo cognitivo de los niños y niñas.

Tabla 1: Evaluación de objetivos Alcanzados.

Tipo	Objetivos	Resultados
General	Desarrollar un videojuego educativo con técnicas de inteligencia artificial para la plataforma Android aplicando la Metodología OOHDM. Caso de Estudio: Laberinto en 3D.	Se desarrolló con éxito el videojuego educativo con IA utilizando las tecnologías mencionadas.
Específico	Realizar el diseño y desarrollo de la aplicación del videojuego con el motor de juegos UNITY aplicando la metodología OOHDM con UML.	La aplicación base fue creada para poder cargar cualquier modelo de laberinto en 3D generados con IA utilizando técnicas de planificación. Se aplicó la Metodología OOHDM junto con UML en el desarrollo lo cual facilitó el control del avance del proyecto.
	Desarrollar el modelo 3D de la aplicación con la herramienta Autodesk Maya 2012.	Se crearon los modelos de las paredes, del piso y el avatar del juego representado por un ratón en 3D, para su posterior utilización en el aplicativo del juego.
	Implementar el movimiento de cámaras para realizar un recorrido virtual por el laberinto en 3D, mediante el uso del motor de juegos UNITY con el lenguaje de programación C# con Mono.	Se implementó un recorrido virtual a fin de cumplir con los requisitos propuestos y es posible generar un laberinto en 3D con IA y resolver el mismo también con IA y almacenar los puntajes tanto del usuario como de la PC en archivos planos tipo XML.

5. TRABAJOS RELACIONADOS

Se han encontrado dos trabajos relevantes de videojuegos educativos conocidos como laberintos: (1) El trabajo propuesto por (Romanovna O., 2009), titulado The Game in C++, está concebido como un laberinto en 2D desarrollado con la herramienta de programación Visual C++ 2008 y el motor de gráficos Simple DirectMedia Library, donde se construyen una serie de laberintos con un avatar que puede ser operado manualmente con las flechas del teclado y evalúa el funcionamiento de las librerías gráficas DirectMedia, pero no utiliza IA para generar los escenarios de los laberintos, ni para resolver los mismos. (2) El trabajo propuesto por (Vander Sterren William, 2001), titulado Terrain Reasoning for 3D Action Games, donde se desarrollan una serie de algoritmos con IA para generar escenarios de juegos para terrenos (terrains) los mismos que pueden ser aplicables para generar escenarios para laberintos y que es un tipo especial de terreno que tiene una entrada y una salida para cumplir con un objetivo.

6. CONCLUSIONES Y TRABAJO FUTURO

El presente trabajo presentó el análisis, diseño e implementación de un videojuego educativo conocido como laberinto en 3D que se ha realizado empleando Autodesk Maya, C# con Mono dentro de la plataforma de desarrollo MonoDevelop y el motor de juegos UNITY como componente base para el proceso de creación gráfica de los videojuegos. En el proceso de creación del software educativo, se aplicó la metodología OOHDM con UML, como un marco de trabajo referencial que se ha cumplido con cada una de las fases desde el análisis, pasando al diseño, hasta llegar al desarrollo e implementación del videojuego. Los resultados muestran la funcionalidad del modelo en 3D del laberinto con las paredes, los pasadizos, el personaje o avatar del juego, lo cual permite mostrar a los usuarios finales el estado real de un laberinto para jugar. El proyecto desarrollado constituye una aplicación piloto para la creación de nuevos videojuegos educativos con IA en beneficio de los niños para estimular el desarrollo cognitivo, que aplicará estos métodos de difusión para dar a conocer los trabajos que se realizan tanto a nivel nacional como a nivel mundial.

Como trabajo futuro se plantea un estudio de los juegos reales que más éxito han tenido hasta la actualidad y que pueden ser simulados en mundos virtuales y en entornos distribuidos.

7. REFERENCIAS BIBLIOGRÁFICAS

Blackman Sue, (2011). Beginning 3D Game Development with Unity, Apress, New York – USA.

Druzhinina Romanovna Olga, (2009). Thesis: The Game in C++. Supervisor: Dr. Tony Y. T. Chan. School of Business and Science University of Akureyri (UNAK). Akureyri – Iceland.

Monedero R., (2013). Tesis de Grado de Ingeniería Informática. Demo de un Videojuego 2.5D en Unity 3D con Blender. Director: Eloi Puertas Prats. Depto, de Matemática Aplicada. Universidad de Barcelona. España.

Pérez M., (2008): Penetración de los videojuegos educativos e infantiles en España desde el 2005 al 2007. *Comunicación y Pedagogía: Nuevas tecnologías y recursos didácticos*, nº 229, pp. 2328.

Pressman Roger, (2002). Ingeniería de Software Un Enfoque Práctico, Mc. Graw Hill, Madrid – España.

Rodriguez E., (2002). Jóvenes y videojuegos. Espacio, significación y conflictos. FAD (Fundación de Ayuda contra la Drogadicción). ISBN: 8495248182. España – Madrid.

Rossi, G.; Schwabe, D.; De Lucena, C.J.P.; Cowan , D.D.; (1995). An Object Oriented Model for Designing the Human Computer Interface of Hypermedia Applications, Proc. of the International Workshop on Hypermedia Design (IWHD'95), Springer Verlag Workshops in Computing Series, forthcoming.

Russell Peter, (2010). Artificial Intelligence. A Modern Approach. 3rd Ed. Pearson Education. New JerseyUSA.

Villacís, C., Fuertes, W., Bustamante, A., Almachi, D., Procel, C., Fuertes, S., & Toulkeridis, T. (2014). Multi player Educational Video Game over Cloud to Stimulate Logical Reasoning of Children. In Proceedings of the 2014 IEEE/ACM 18th International Symposium on Distributed Simulation and Real Time Applications (pp. 129 137). IEEE Computer Society.

Torres Vinueza, M. D., Fuertes, W., Villacís Silva, C. X., Zambrano Rivera, M. E., & Prócel Silva, C. T. (2013). Puzzlemote: videojuego controlado con el mando de la Wii para niños de 6 a 10 años. AtoZ: novas práticas em informação e conhecimento, 2(2), 94105.

Vander S. William, (2001). Terrain Reasoning for 3D Action Games. Gama Sutra Magazine. UBM Tech. USA.

Loh Peter K. K. and Prakash Edmond C, (2009). Performance Simulations of Moving Target Search Algorithms. Hindawi Publishing Corporation. International Journal of Computer Games Technology. Volume 2009, Article ID 745219,